DTU Library

Heuristic and exact algorithms for vehicle routing problems

Røpke, Stefan

Publication date:
2006

Heuristic and exact algorithms for vehicle routing problems

Stefan Ropke

December 2005

Preface

This Ph.D. thesis has been prepared at the Department of Computer Science at the University of Copenhagen (DIKU), during the period November 2002 to December 2005. The work has been supervised by Professor David Pisinger.

The thesis consists of four introductory chapters: Chapters 1, 2, 3 and 7, and five research papers: Chapters 4, 5, 6, 8 and 9. The five research papers have been written in collaboration with coauthors which are mentioned in the beginning of each paper. The four introductory chapter have been written solely by the undersigned. The five research papers are relatively self-contained. Note that each research paper contains it own bibliography and sometimes an appendix. The bibliography for the introductory chapters are found at the end of this thesis.

The thesis contains three parts. The first part contains the introduction and is split into two chapters. The next part deals with heuristic and contains one introductory chapter and three research papers about heuristics. These papers are "technical report versions" that contains more results than the papers that have been submitted to journals. These extra results are placed in the appendix of each paper. The last part is about exact methods and contains one introductory chapter and two research papers.

The thesis started out being solely about heuristics, but after having worked with heuristics for four or five years, first as a graduate student, then in the industry and as a Ph.D. student I felt it was time to learn something new and started studying exact methods more intensively in 2004. This has certainly been interesting and I hope the knowledge I have obtained will allow me to design even better heuristics in the future.

Chapter 9 is the only one of the five papers that has not been submitted to a journal yet. In its current state it is not ready to be submitted either - it is clearly too long and contains too much material. We do plan to submit a condensed version. The rest of this section is going to describe how the paper could be condensed. To understand this, one needs to have read chapter 9.

One way of condensing the paper would be to focus on the SP1 and SP2 relaxations and leave the SP3 and SP4 relaxations out as well as the addition of valid inequalities. The contribution of this paper would be

1. Improvements of domination criteria for ESPPTWCPD.
2. The computational comparison of SP1 and SP2.
3. The new pricing heuristics and experiments. More experiments could be carried out.
4. Introduction of standard test instances for exact solution of the PDPTW.

For this paper it would be nice if the issues with algorithms SP1* and SP2* were worked out. The simplest way of doing this would be to use algorithms SP1*/SP2* to get a lower bound. If the linear relaxation solution turns out to be fractional then one should switch to algorithms SP1/SP2 to perform branching.

A better approach would be to implement a branching rule that is compatible with the strongest domination criteria. Branching on time windows as proposed in the paper would be a good candidate. An alternative is to find a way of perturbing the (d_{ij}) matrix such that $d_{ij} + d_{jk} \geq d_{ik}$

always holds when j is a delivery node, even when general cuts have been added to the master problem. Valid perturbations of the (d_{ij}) matrix include subtracting a constant α_i from all edges leaving pickup node i and adding α_i to all edges leaving node $n+i$. This is valid as a path in the ESPPTWCPD and SPPTWCPD that visits a pickup must visit the corresponding delivery and vice versa. This would allow us to add cuts in the original variables to the master problem and would thereby make the current branching rule work with with SP1*/SP2*.

A second paper could describe the SP3 and SP4 relaxations and incorporate the valid inequalities in the branch-and-price algorithm. This paper could also include the strengthened SP4 relaxation that is described in the conclusion of the paper.

Acknowledgments

I would first of all like to thank my supervisor, Professor David Pisinger for encouragement, countless discusions and for his help with writing the thesis. Without David I would not had taken on the task of doing a Ph.D. study. Associate Professor Jean-François Cordeau and Professor Gilbert Laporte also deserves great thank for making my visits to the University of Montreal possible and for taking time to work with me while I have been visiting. The input I received from my advisory group, Professor Jacques Desrosiers and Professor Oli Madsen, is also greatly appreciated.

I would also like to thank the guys at the Algorithmics and Optimization Group at DIKU for encouragement and for making the average work day more fun and interesting. Similarly I would like to thank the people I met at the Centre for Research and Transportation in Montreal, especially “the gang”, for making me feel welcome in a foreign country. I also wish to thank Irina Dumitrescu for her patience with me when I have postponed working on our joint projects because of the work involved in finishing this thesis.

Finally I would like to thank friends and family for their support. I especially wish to thank my parents for their love and support throughout my life. And to my girlfriend Alice: Thank you for lifting my mood on the days when I have been feeling down in the last couple of months, for helping me improving the language in the thesis and for being you!

Copenhagen, December 2005, Stefan Røpke

Updated version, June 2006

A number of typos and errors have been corrected in this version of the thesis. Since December 2005 I have spent time working on the research paper presented in chapter 9. This work has lead to resolution of the most of the issues discussed above and mentioned in the chapter 9: A transformation of the distance matrix has been found that makes it possible to use SP1*/SP2* after adding cuts in the master problem and it has been shown that many of the cuts that seemed worthless in the computational experiements in fact are implied by the strongest set partition relaxations. These developments have not been included in the updated version of the thesis, but are described in Ropke and Cordeau [2006].

Let me use the opportunity to thank my opponents: Stefan Irnich, Daniele Vigo and Martin Zachariasen at the Ph.D. defense, for evaluating the thesis within a short time and for valuable comments that has lead to several improvements in this updated version of the thesis.

Montreal, June 2006, Stefan Røpke

Contents

Part I

Introduction

Chapter 1

Introduction

1.1 Motivation

Transportation of goods and passengers is an important task in the society of today. Astronomical amounts of money are spent daily on fuel, equipment, maintenance of equipment and wages.

It is therefore obvious to attempt to reduce the amount of money spent on transportation as even small improvements can lead to huge improvements in absolute terms. Several approaches could be taken, one could improve equipment or make the infrastructure better. One could also look at operations research (OR) techniques - given the resources available, what is the best that can be done? Toth and Vigo [2002b] estimate that the use of computerized procedures for planning of the distribution process often leads to savings in the area of 5% to 20% of the transportation costs, so studying such procedures definitely seems worthwhile.

Furthermore, transportation accounts for a great part of the CO_2 pollution in the world today. According to Pedersen [2005] the transportation sector was in 1998 responsible for 28% of the CO_2 emission in the Europe Union and road transportation alone accounted for 84% of the total CO_2 emissions from the transportation sector. Moreover, it is exected that the CO_2 emissions from the transportation sector is going to increase by 50% by 2010 (according to Pedersen [2005]). Thus, improvements in planning techniques could help easing the strain on the environment caused by transportation.

Operations research has been quite successful in the transportation area. One could see optimization within transportation as one of the successes of OR. Today OR techniques are applied within for example the airline, railway, trucking and shipping industries; and OR techniques are used to optimize the interplay between the different modes of transportation, for example in handling port operations.

Several companies exist that solely or primarily develop software for optimization within the transportation industry. Some examples are the Swedish based *Carmen Systems*[1] that develop software for airlines and railways; the Canadian *Giro*[2] that develops software for routing and scheduling of ground based vehicles; the Danish *Transvision*[3] that develops software for ground based distribution; the Danish *e2e factory*[4] that develops software controlling ground personnel at airports.

Consequently it is fair to say that optimization within transportation is a subject that is used and sought-after in the real world and not just a topic studied in academia. Also, the field seems to have reached a certain level of maturity as it has been studied for many decades. Having said that, there remain ample room for improvement in the solution methods employed and OR methods could be applied to a wider array of problems faced within the transportation industry. The real world need solution methods that are:

[1] http://www.carmen.se
[2] http://www.giro.ca/
[3] http://www.transvision.dk
[4] http://www.e2efactory.dk/

- Fast — the quicker the operator gets an answer back from the computer the better,
- Easy to apply to a variety of problem characteristics — when developing software for real life problems one wants to avoid reinventing the wheel every time a new client wants a software application for a new type of transportation problem,
- Precise — the better results a solution method returns the larger is the potential for savings,
- More robust — when solving real world problems it is often better to have a solution method that produces fairly good results for all problem instances, than one that produces very good results for 70% of the problem instances and very poor results for the remaining 30%.

The four characteristics listed above are to a certain extent in conflict with each other, so some sort of trade-off has to be achieved. Solution methods described in the literature are often evaluated in terms of speed, solution quality, and to a certain extent, robustness while the second characteristic listed above often receives less attention. In this thesis a solution method that takes all four characteristics into account is presented.

One problem in the field of transportation related OR that has been given a lot of attention in the scientific literature is the so called *vehicle routing problem* (VRP). In the vehicle routing problem we are given a fleet of *vehicles* and a set of *customers* to be visited. The vehicles are often assumed to have a common home base, called the *depot*. The cost of traveling between each pair of customers and between the depot and each customer is given. Our task is to find a route for each vehicle, starting and ending at the depot, such that all customers are served by exactly one vehicle, and such that the overall cost of the routes are minimized. Typically the solution has to obey several other restrictions, such as capacity of the vehicles or desired visit times at customers. In this thesis the term *vehicle routing problem* (VRP) is used to describe a broad class of problems and not a specific problem with a specific set of restrictions or constraints. The class of vehicle routing problems contains all the problems that involve creating one or more routes, starting and ending in one or more common depots or at predefined start and end terminals. In the literature the term vehicle routing problem is occasionaly used for the specific problem that is called the *capacitated vehicle routing problem* in this thesis (see Section 2.3).

A subclass of vehicle routing problems is *pickup and delivery problems*. In this class of problems we are given a number of *requests* and a fleet of vehicles to serve the request. Each request consists of a pickup at some location and a delivery at another location. The cost of travelling between each pair of locations is given. The problem is to find routes for each vehicle such that all pickups and deliveries are served and such that the pickup and delivery corresponding to one request is served by the same vehicle and the pickup is served before the delivery. Again a number of additional constraints are often enforced, the most typical being capacity and time window constraints. Figure 1.1 shows how transportation problems, vehicle routing problems and pickup and delivery problems relate to each other. Pickup and delivery problems are shown as the innermost, most specialized problem class, but it contains many classical vehicle routing problems like the *capacitated vehicle routing problem* (CVRP) and the *vehicle routing problem with time windows* (VRPTW). How these and many other vehicle routing problems can be formulated using one pickup and delivery model is discussed in Chapter 4–6. The *pickup and delivery problem with time windows* (PDPTW) is the core problem studied in this thesis. In Chapter 2 the problem is formally defined together with some of the classic problems it generalizes.

1.2 Modeling and solution methods

The research within an area like vehicle routing problems can be grouped into two major categories: modeling and solution methods. A third area of interest is the interpretation of the results stemming from the models and solution methods. This area is typically studied together with either modeling or solution methods. The following two sections go into further details with modeling and solution methods.

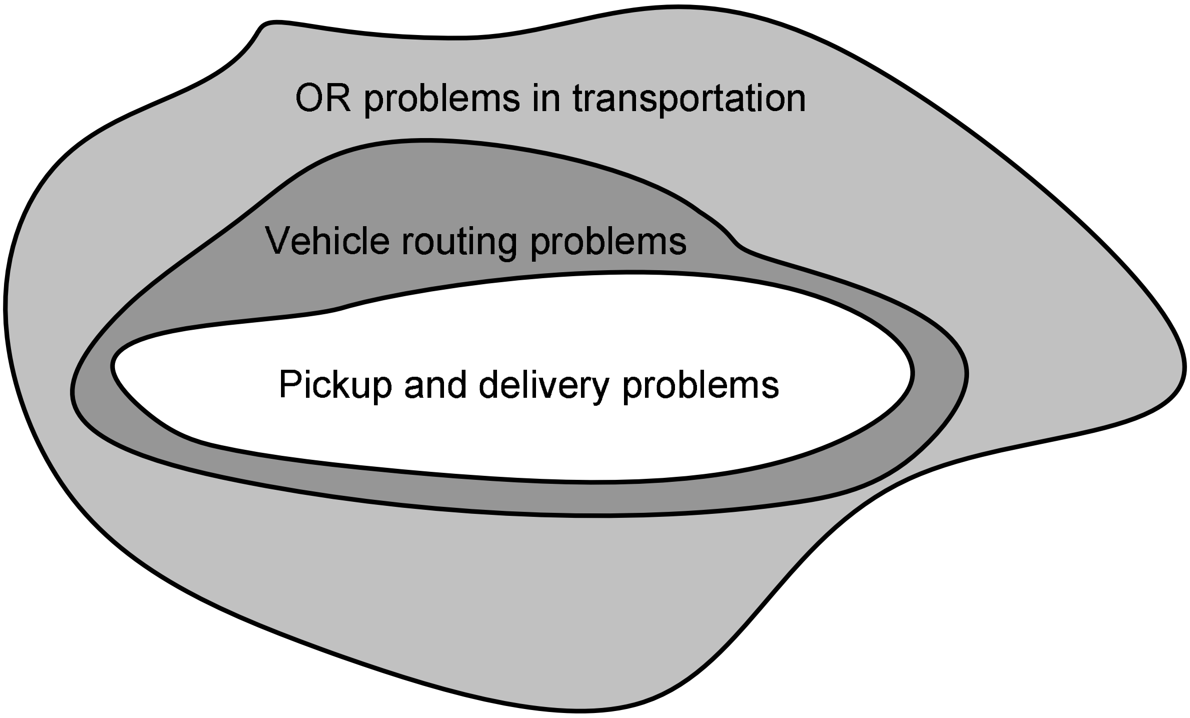


Figure 1.1: The figure shows that the vehicle routing problems is one of many problems studied within operations research applied to transportation problems. Pickup and delivery problems are a subclass of vehicle routing problems. The class of transportation related OR problems, of course contains many other problems apart from vehicle routing problems. Some examples are train timetabling problems [Caprara et al. [2002]] or berth allocation problems [Imai et al. [2003]].

1.2.1 Modeling

The art and science of modeling can be roughly divided into two disciplines. The first discipline is concerned with modeling a problem occurring in real life. The following topics must be considered:

- While the description of the real life problem given to the modeler, may be vague and ambiguous, the opposite should be true for a good model. A good model should be expressed such that there are no ambiguities - everyone (with the right qualifications) who reads the model should get the same idea of what the model represents. This can be achieved by using a mathematical notation, but a textual representation can be sufficient as well. Notice that a mathematical model does not guarantee that the model is without ambiguities.
- The model must represent the real life problem reasonably well. The word *reasonably* is vague, but how close to the real life problem the model should be is dependent on the application. Often we do not want to model the real life situation in all its details for different reasons, one such reason could be that precise data is missing and another is given in the next bullet point.
- The model should not be unnecessarily complicated. As we often want to solve the problem using a computer program the model should be manageable - it might be necessary to leave out some details of the real life problem to make the model solvable by the methods we know today.

How to model a real life problem is a very important but also quite challenging task. Furthermore it may be difficult to decide if a model is good or not, or to choose between two different models that are supposed to represent the same problem. Such a decision can be dependent on experience and personal preferences.

The second discipline in modeling is how to transform one model into an equivalent model that either in some way is easier to solve using existing techniques or paradigms or that makes the model solvable using a particular tool. The word *equivalent* should be understood in a strict sense. The new model should have the same solution as the original model given the same input. An example could be the reformulation of an integer programming model to another model that provides a tighter linear relaxation and consequently might be better in a linear programming based branch and bound algorithm.

When transforming one model into another, the underlying modeling framework we are transforming to is important - the more expressive and rich it is, the easier the modeling becomes.

This thesis contains many examples of the second modeling discipline. Chapter 5 and 6 show how many commonly studied vehicle routing problems can be reformulated into a pickup and delivery problem and solved using the tool (heuristic) developed in Chapter 4. Chapters 8 and 9 propose different models for the PDPTW and evaluate which model that is best suited as the basis of an exact algorithm for the PDPTW.

This thesis does not explicitly deal with the first modeling discipline. This does not mean that the thesis is uninteresting for practitioners, working with real world problems though. The heuristic developed in the first three papers (Chapter 4 to 6) is able to handle a variety of constraints and is therefore better suited for application to real life problems compared to many special purpose heuristic proposed in the scientific literature. A variant of the heuristic is actually used in practice by at least one company solving real life problems.

1.2.2 Solution methods

For many of the problems considered in this thesis, the set of feasible solutions is so large that even if we had a computer that in a systematic way could construct and evaluate the cost of a trillion (10^{12}) solutions per second, and we had started that computer right after the big bang, 14 billion years ago, it would still not have evaluated all the feasible solutions today. Consequently we have to turn to other methods than simple enumeration.

Three types of solution methods are typically employed to solve these types of problems (NP-hard problems):

- **Heuristics.** Heuristics are solution methods that typically relatively quickly can find a feasible solution with reasonable quality. There are no guarantees about the solution quality though, it can be arbitrarely bad. The heuristics are tested empirically and based on these experiments comments about the quality of the heuristic can be made. Heuristics are typically used for solving real life problems because of their speed and their ability to handle large instances.
  A special class of heuristics that has received special attention in the last two decades is the *metaheuristics*. Metaheuristics provides general frameworks for heuristics that can be applied to many problem classes. High solution quality is often obtained using metaheuristics. Part II of thesis is concerning heuristics.

- **Approximation algorithms**. Approximation algorithms are a special class of heuristic that provide a solution **and** an error guarantee. For example one method could guarantee that the solution obtained is at most k times more costly than the best solution obtainable. Two classes of approximation algorithms called *polynomial time approximation scheme* (PTAS) and *fully polynomial time approximation scheme* (FPTAS) are of special interest as they can approximate the solution with any desired precision. That is for any instance I of the problem considered and any $\epsilon > 0$ a PTAS or FPTAS can output a solution s such that $f(s) \leq (1+\epsilon)Opt$ (assuming that we are solving a minimization problem) where Opt is the optimal solution and $f(s)$ is objective of solution s. The difference between a PTAS and a FPTAS is that the PTAS is polynomial in the size of the instance I while the FPTAS is polynomial in the size of the instance I and $1/\epsilon$. An FPTAS is therefore in a certain sense "stronger" than a PTAS. An example of a problem that admits an FPTAS is the *Knapsack problem* (see e.g. Kellerer et al. [2004]). For some problems it is not possible to design a FPTAS, PTAS or even an polynomial time approximation algorithm with constant error guarantee unless $P = NP$ and approximation can be impractical: the error guarantee can be too poor or the running time of the algorithm can be too high.
  This thesis is not going to discuss approximation algorithms in further details, we refer the interested reader to Vazarani [2001].

- **Exact methods**. Exact methods guarantee that the optimal solution is found if the method is given sufficiently time and space. As stated initially, a simple enumeration is out of the question, so exact methods must use more clever techniques. The worst case running time for NP-Hard problems are still going to be high though. We cannot expect to construct exact algorithms that solve NP-hard problems in polynomial time unless NP = P. For some classes of problems there are hope of finding algorithms that solve problem instances occuring in practice in reasonable time though.
  Part III of the thesis is concerning exact methods.

1.3 Goals

The focus of this Ph.D. thesis is solution methods for vehicle routing problems and especially pickup and delivery problems. The problems studied in this thesis have been inspired from real world applications but the problems are not real world problems themselves. It is my hope that practitioners can apply some of the solution methods described in this thesis to the problems that occur in real life. This hope has to some extent already been fulfilled.

The thesis is divided into two major parts, one concerning heuristics and one concerning exact methods. In the heuristic part, the focus has been on developing a unified heuristic that is able to handle many of the VRP variants that have been proposed in the literature without any need for retuning the algorithm for a particular problem type.

The research into a unified heuristic for vehicle routing problems led us to investigate robust heuristics in general — is it possible to distill the components of the vehicle routing heuristic into a general heuristic?

The research in exact methods has focused on the pickup and delivery problem with time windows (see Section 2.5). The overall goal of this research has been to push the limits for what sizes of PDPTW problems that can be solved to optimality. In order to do this it has been necessary to investigate new formulations of the problem, preprocessing techniques and valid inequalities.

1.3.1 Achievements and contributions of the Ph.D. thesis

The papers presented in Chapters 4 to 6 describe a general heuristic that successfully handles 12 variants of the vehicle routing problem. The heuristic is able to solve the different problems types without retuning the parameters of the algorithm. The heuristic is able to solve the many variants by first transforming them to a PDPTW instance and then solving that instance using a PDPTW heuristic. For most of the problems the transformation is simple, but the thesis nevertheless presents these transformations for the first time. The heuristic has provided excellent results and has improved the best known solutions to benchmark cases for many problems.

The heuristic has been distilled into a general framework that builds upon the *large neighborhood search* (LNS) paradigm introduced by Shaw [1998]. We call this heuristic framework the *adaptive large neighborhood search* (ALNS). The heuristic is first presented in Chapter 4 which provides an easy to understand description of the ALNS. The chapter also establishes the advantages of ALNS over LNS through computational experiments and presents results on the PDPTW. These results show that the ALNS method must be considered as the best heuristic for the PDPTW currently.

In Chapter 5 the heuristic is applied to a large class of vehicle routing problem with backhauls. A total of 6 variants are considered. For all problem types the heuristic must be considered to be on par with existing specialized algorithms or even better. Some enhancements of the heuristic is proposed and the effect of these enhancements are quantified in computational experiments.

As we believe that the ALNS framework is quite robust and easy to understand and implement, we hope that it can be used outside the vehicle routing domain as well. Consequently, in Chapter 6 we describe the framework in general terms. Also in Chapter 6 we illustrate how a typical search behaves in a novel, graphical way. This leads to a better understanding of how the heuristic explores the solution space and could be used to analyse other metaheuristics as well. The heuristic is tested on 5 new classes of VRPs in Chapter 6, including some classical vehicle routing problems like the capacticated vehicles routing problem and the vehicle routing problem, again with promising results.

The unified heuristic is not only well-suited for solving different types of VRPs but it can also be used to solve problems where a variety of different constraints are in use. The heuristic could for example easily handle a problem where some customers require deliveries from a common depot while other customers need to have goods transported from one location to another.

An implementation of the ALNS heuristic is currently being used to solve real life problems at several large companies in Denmark, so the heuristic has had an impact on real life transportation problems.

In the study of exact methods for the PDPTW two new formulations of the problem have been proposed in Chapter 8. The formulations contain a polynomial number of variables and an exponential number of constraints (in n, the number of requests). These formulations are used in a branch and cut approach to solve the problem to optimality. The computational experiments show that the new formulations enable us to solve much larger instances to optimality compared to an earlier branch and cut algorithm by Cordeau [2006]. Furthermore two new classes of valid inequalities are presented, the so called *strengthened capacity inequalities* and *fork inequalities.* Heuristic separation procedures for the two classes of inequalities are also presented. The last class of inequalities proves to be especially helpful in increasing the lower bound of the LP relaxation. The last contribution in Chapter 8 is the adaptation of the so called *reachability inequality,* introduced for the VRPTW by Lysgaard [2005], to the PDPTW.

Chapter 9 compares several lower bounds obtained by solving the set-partitioning formulation of the PDPTW by column generation using different pricing problems. Two pricing problems from the literature are investigated and two pricing problems that has not been used for the PDPTW before are proposed . This provides the first computational comparison of the lower bounds obtained by using the pricing problem proposed by Dumas et al. [1991] to the lower bound obtained by Sol [1994].

The lower bound obtained by solving the set partitioning relaxation is strengthened by adding valid inequalities, thus the implemented algorithm is of the branch-cut-and-price type. The chapter also introduces a new valid inequality for the PDPTW, the *strengthened precedence inequality.* This inequality is obtained by combining the ideas of the reachability inequality mentioned above with the precedence inequality proposed by Ruland and Rodin [1997].

The computational results show that the branch-and-cut-and-price algorithm is able to outperform the branch and cut algorithm from Chapter 8 on most instances considered in the test.

1.4 Overview of Ph.D. thesis

The thesis is divided into two parts, a part about heuristics and a part about exact methods. Each part begins with a short introduction to the field and the papers contained in that part. The heuristic part contains three papers:

- **Chapter 4: An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows**. This paper presents the adaptive large neighborhood search (ALNS) heuristic and applies it to the PDPTW. The paper is concluded with a computational experiment that shows the superiority of ALNS to a simpler larger neighborhood search (LNS). The results on standard benchmark problems for the PDPTW show that the heuristic overall obtain the best results compared to competing heuristics. The paper has been accepted for publication in Transportation Science and it is co-authored with David Pisinger.
- **Chapter 5: A unified heuristic for a large class of vehicle routing problems with backhauls**. This paper uses the heuristic from the preceding chapter to solve a large class of vehicle routing problems with backhauls. The paper gives a short survey of vehicle routing problems with backhauls and describes how the problems can be transformed to the PDPTW. On the algorithmic side the paper suggests some improvements to the ALNS. The computational tests show that the improvements suggested in the paper do have a positive impact on the algorithm and they show that the heuristics must be considered to be at least on par with existing, specialized heuristic for the considered problems when it comes to solution quality. The paper has been accepted for publication in a special issue of European Journal of Operational Research and it is co-authored with David Pisinger. This paper was submitted before the paper in Chapter 4, and consequently uses a slightly different vocabulary. Most notable is it, that the term ALNS is not used in this paper.
- **Chapter 6: A general heuristic for vehicle routing problems**. This paper gives a general description of the ALNS framework as we believe it can be applied to optimization problems outside the vehicle routing domain. The paper extends the unified vehicle routing heuristic to handle five additional problem classes, including the CVRP, VRPTW and MDVRP (multi depot VRP). The paper is concluded with computational experiments that among other things show that the unified heuristic is the best method currently, when it comes to minimizing the number of vehicles in large VRPTW instances. The paper has been accepted for publication in Computers and Operations Research and is co-authored with David Pisinger.

The part about exact methods contains two papers

- **Chapter 8: Models and a Branch-and-Cut Algorithm for Pickup and Delivery Problems with Time Windows**. This paper proposes two new models for the PDPTW, both models contains an exponential number of constraints. These constraints are added dynamically to the model along with other valid inequalities. The paper proposes two new classes of valid inequalities, the so called *fork inequalities* and *strengthened capacity inequalities*. An inequality recently proposed for the VRPTW, the so called *reachability inequality* is also adapted to the PDPTW. Computational experiments show that the new formulations are superior to a formulation proposed recently by Cordeau [2006], for some instances a speedup of more than a factor 1000 is observed. The paper has been submitted to a special issue of Networks and has been conditionally accepted. It is co-authored with Jean-François Cordeau and Gilbert Laporte.

- **Chapter 9: Branch-and-Cut-and-Price for the Pickup and Delivery Problem with Time Windows**. This paper examines the set-partitioning formulation (see for example Dumas et al. [1991]) for the PDPTW. Four different relaxations of the problem are proposed by varying the pricing problem in a column generation algorithm for the problem. Two of these pricing problems have previously been considered as pricing problems for the PDPTW. This paper gives the first computational comparison of the two lower bounds obtained by using these two pricing problems and it improves upon the exact algorithms for one of the problems by improving the dominance criterion. Valid inequalities proposed in Chapter 8 are added to the model dynamically and a new class of valid inequalities is proposed, which is denoted the *strengthened precedence inequality*. Extensive computational results show that the branch-cut-and-price algorithm usually is superior to the branch-and-cut algorithm, but that this is not always is the case. The paper is co-authored with Jean-François Cordeau and has not yet been submitted. Plans for publication were discussed in the preface.

The three papers about the heuristic have been presented in different forms at the following occasions

- *Route2003* - International Workshop on Vehicle Routing, Denmark, June 22-25, 2003 (speaker: Stefan Ropke).

- *The EURO Summer Institute - ESI XXI Stochastic and Heuristic Methods in Optimization,* July 25 - August 7 2003, Neringa, Lithuania (speaker: Stefan Ropke).

- *ISMP2003* - International Symposium on Mathematical Programming, Denmark, August 18-22, 2003 (speaker: Stefan Ropke).

- *CORS/INFORMS International Meeting,* Banff 2004, May 16-19, 2004 (speaker: Stefan Ropke).

- Seminar at the Center for Research on Transportation, University of Montreal, Canada, September 30, 2004 (speaker: Stefan Ropke).

- *Route2005* - International workshop on vehicle routing and intermodal transportation, Bertinoro, Italy - June 23-26, 2005 (speaker: David Pisinger)

Chapter 8 has been presented at the following occasions

- *International Colloquium for the 25th anniversary of GERAD*, Montreal, Canada, May 11-13, 2005, (preliminary version) (speaker: Jean-François Cordeau)

- *Route2005* - International workshop on vehicle routing and intermodal transportation, Bertinoro, Italy - June 23-26, 2005 (speaker: Jean-François Cordeau)

- Seminar at Mathematics department, Brunel University, 19th December 2005 (planned) (speaker: Gilbert Laporte)

A very early version of Chapter 9 was presented at

- Optimization days 2005, Montreal, Canada, May 9-11, 2005 (speaker: Stefan Ropke)

Chapter 2

Classes of vehicle routing problems

The objective of this section is to introduce the core problem studied in this thesis, *the pickup and delivery problem with time windows* (PDPTW). In order to do so four simpler variants of the problem are first introduced. This gives an introduction to some of the core problems in the field of vehicle routing and surveys the relevant literature. The four preliminary problems studied are the *traveling salesman problem* (Section 2.1), the *m-traveling salesmen problem* (Section 2.2), the *capacitated vehicle routing problem* (Section 2.3) and the *vehicle routing problem with time windows* (Section 2.4). The section on the *pickup and delivery problem with time windows* can be found in Section 2.5. Each section first introduces the problem in words and then gives a mathematical definition of the problem. Finally literature pointers are given and recent advances are discussed (in Section 2.3, 2.4 and 2.5). The introduction of the problem and mathematical models can be understood with a basic knowledge of operations research, while the literature discussion can be technical at times and requires a deeper understanding of operations research and in particular of solution method paradigms.

It should be mentioned that all five problem classes discussed here are NP-hard.

2.1 The traveling salesman problem

One of the simplest, but still NP-hard, routing problems is probably the *traveling salesman problem* (TSP). In the TSP one is given a set of cities and a way of measuring the distance between each city. One has to find the shortest tour that visits all cities exactly once and returns back to the starting node. In Figure 2.1 an example of a TSP instance is shown to the left and to the right the optimal solution is shown when Euclidean distances are used to measure the distance between two cities.

The problem comes in different flavours depending on what properties the distances satisfy. If the distances satisfy that the distance from city i to city j is the same as the distance from city j to city i for all cities i and j, the the problem is said to be symmetric. If this property does not hold then the problem is said to be asymmetric. A problem is said to be Euclidean if the cities are located in $\mathbb{R}^d$ and the distance between two cities is the Euclidean distance.

The problem can be formulated as a mathematical model in the following way. Let $G = (V, A)$ be a complete, directed graph where $V = \{1, \dots, n\}$ is the set of nodes/cities and A is the set of arcs. To each arc $(i, j) \in A$ is a assigned a distance or cost c_{ij}. We define binary decision variable x_{ij} that is set to one if and only if arc (i, j) is used in the solution. The problem can be formulated as

$$\min \sum_{i \in V} \sum_{j \in V \setminus \{i\}} c_{ij} x_{ij} \tag{2.1}$$

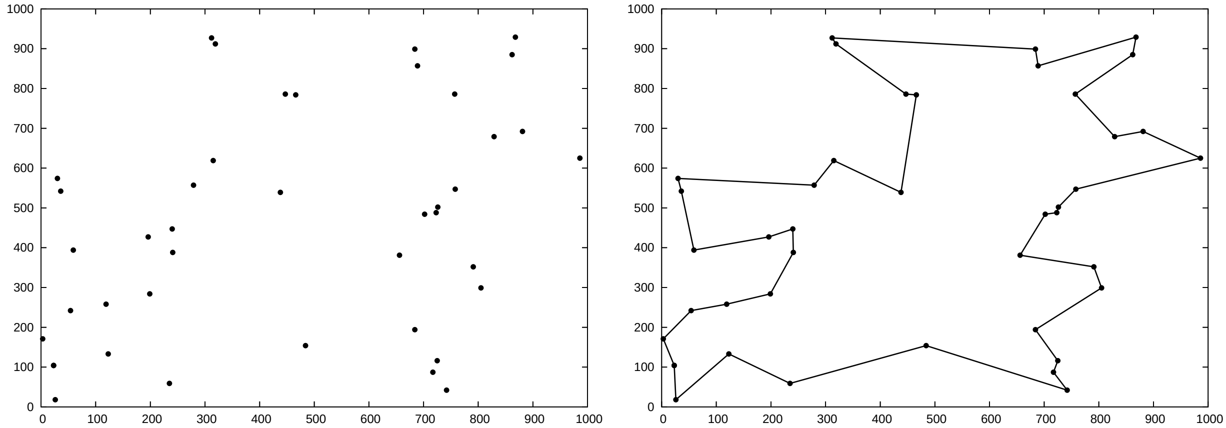


Figure 2.1: TSP illustration

subject to

$$\sum_{j \in V \setminus \{i\}} x_{ij} = 1 \qquad \forall i \in V \tag{2.2}$$

$$\sum_{i \in V \setminus \{j\}} x_{ij} = 1 \qquad \forall j \in V \tag{2.3}$$

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq 1 \qquad \forall S \subset V \tag{2.4}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A \tag{2.5}$$

The objective (2.1) minimizes the arc costs, equations (2.2) and (2.3) ensures that one arc leaves each node and one arc enters each node, equation (2.4) eliminates sub-tours.

The amount of scientific literature on the TSP is staggering. Good starting points for getting to know the problem are E. L. Lawler and Shmoys [1985] and Gutin and Punnen [2002]. The origins of the TSP are discussed in Schrijver [2005]. Very large Euclidean instances of the TSP can be solved to optimality, the largest instance solved to optimality so far contains 24,978 cities. It was solved by branch-and-cut by the research team of Applegate, Bixby, Cvátal, Cook and Helsgaun[1]. Heuristic methods for the TSP have been applied to an instance with more than 1.9 million cities and the gap between the currently best know upper and lower bounds for this instance has been shown to be 0.068%[2] which is quite remarkable. It is safe to say that the TSP is one of the most studied NP-hard problems and solution methods for this problem have reached a very high level. More general routing problems like the capacitated vehicle routing problem or the pickup and delivery problem with time windows turn out to be much harder to solve, both heuristically and exactly, compared to the TSP. I think that the impressive development in solution methods for the TSP leaves hope of significant improvements in solution methods for the more general routing problems.

2.2 m-Traveling salesman problem

The *m-traveling salesman problem* (m-TSP) is a generalization of the TSP that introduces more than one salesman. In the m-TSP we are given n cities, m salesmen and one *depot* or *home base*. All cities should be visited exactly once on one of m tours, starting and ending at the depot. The tours are not allowed to be empty. If distances satisfy the triangle inequality, that is

[1] http://www.tsp.gatech.edu/sweden/index.html
[2] http://www.tsp.gatech.edu/world/index.html

if $d(i,k) \leq d(i,j) + d(j,k)$ for all i, j and k then it is easy to see that the distance of the shortest TSP tour on the n cities plus the depot always is less than or equal to the distance of the shortest m-TSP solution for any m.

Any m-TSP with n cities can be formulated as a TSP with $m + n$ cities. One first creates m copies of the depot node. The distances between depot nodes is then set to a sufficiently large number while the distances between the depot nodes and ordinary nodes are copied from the m-TSP. The large distance between depot nodes ensures that no salesmen tours are empty. Notice that the resulting TSP does not obey the triangle inequality.

In Figure 2.2 an example of a solution to an $m - TSP$ with $m = 3$ and $n = 38$ is shown. The actual solution is shown in the top right part of the figure. Observe that one salesman serves one city, the next salesman serves 4 cities and the last salesman serves the rest of the cities. Thus the workload is by no means split fairly between the salesmen.

The m-TSP is not studied widely in the literature, probably because it is so closely related to the TSP. The literature about heuristics and exact methods has recently been surveyed by Bektas [2006]. An interesting variant of the problem is the min-max m-TSP where the length of the longest salesman tour has to be minimized. This problem has been studied by França et al. [1995] who proposed heuristic and exact methods for the problem. More recently Applegate et al. [2002] solved a challenging min-max m-TSP instance to optimality for the first time. The instance originated from a competition from 1996 and had been unsolved since then. The problem was solved on a network of 188 processors and required 10 days of computing, which corresponds to roughly 79×10^6 CPU seconds scaled to a 500 MHz Alpha EV6 processor.

2.3 Capacitated vehicle routing problem

In the *capacitated vehicle routing problem* (CVRP) a vocabulary different from the one used in the TSP community is used. The objects called cities in the TSP world are called *customers* in the CVRP world and the salesmen are called *vehicles*. The common starting point is still denoted the *depot*. In the CVRP we are given a depot, a set of n customers, a set of m vehicles and a distance measure as in the m-TSP, but in the CVRP every vehicle has a *capacity* Q and every customer $i \in \{1, \ldots, n\}$ has a *demand* q_i. The task in the CVRP is to construct vehicle routes such that all customers are served exactly once and such that the capacities of the vehicles are obeyed. This should be done while minimizing the total distance traveled.

We now introduce a mathematical model for the problem. We use a set partitioning approach (or path-based modeling) as this makes it easier to model the more complicated problems that are described below. A model similar to the one presented for the TSP (Section 2.1) is certainly possible; such a model can be found in Toth and Vigo [2002b]. Let $G = (V, A)$ be a directed graph as before, let $V = \{0, 1, \ldots, n, n+1\}$ be the set of nodes in the graph where node 0 and $n + 1$ corresponds to the depot and node $\{1, \ldots, n\}$ corresponds to customers. The depot has been split into two nodes to make modeling easier, node 0 corresponds to the start of the routes and $n + 1$ corresponds to the end of the routes. We assume that distances are given as a matrix $(c_{ij}), i, j \in \{0, \ldots, n + 1\}$. From now on we will call the distances for *costs*.

A legal route $\bar{r}$ must be a simple (that is, no node is visited twice) path from node 0 to node $n + 1$. We can write such a path

$$\bar{r} = (v_0, v_1 \ldots, v_h, v_{h+1}) \tag{2.6}$$

where v_i, $i \in \{0, \ldots, h + 1\}$ are the nodes visited on the route. We always have that $v_0 = 0$ and $v_{h+1} = n + 1$. h is the number of customers visited on the route. The route should satisfy the capacity requirement. We can write this as

$$\sum_{i=1}^{h} q_{v_i} \leq Q \tag{2.7}$$

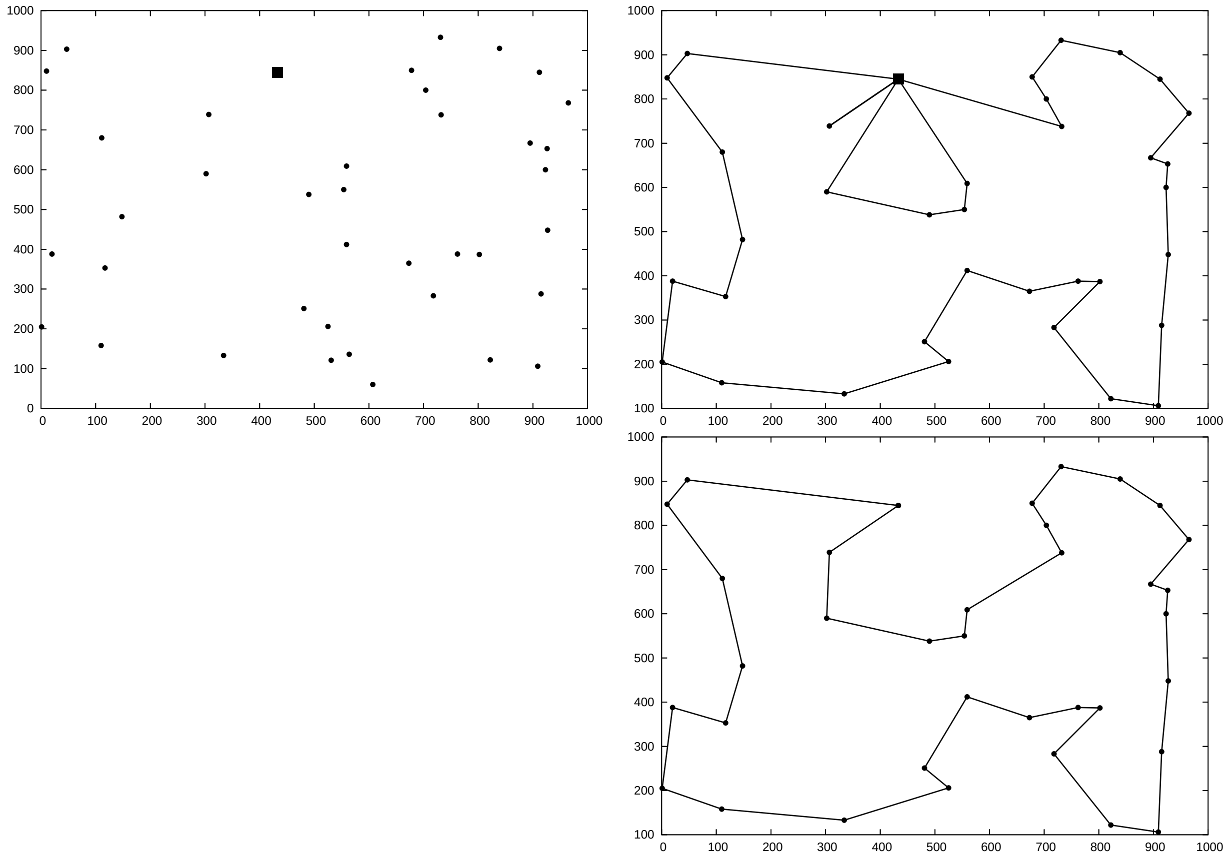


Figure 2.2: m-TSP figure. Top left: the depot and cities in the instance. The depot is indicated as a black square. Top right: the optimal m-TSP solution for $m = 3$. Bottom right: The optimal TSP solution when the TSP was solved on the same set of cities plus the depot. The length of the m-TSP solution is 5699 units while the length of the TSP solution is 5026 units.

the cost $c_{\bar{r}}$ of a route $\bar{r}$ is

$$c_{\bar{r}} = \sum_{i=0}^{h} c_{v_i,v_{i+1}} \tag{2.8}$$

Let R be the set of all feasible routes and let $(a_{i\bar{r}})$ be a boolean matrix with n rows and $|R|$ columns. Let $a_{i\bar{r}} = 1$ if and only if route $\bar{r}$ serves customer i. The CVRP can be formulated as

$$\min \sum_{\bar{r}\in R} c_{\bar{r}} x_{\bar{r}} \tag{2.9}$$

subject to

$$\sum_{\bar{r}\in R} a_{i\bar{r}} x_{\bar{r}} = \quad 1 \qquad \forall i \in \{1, \ldots, n\} \tag{2.10}$$

$$\sum_{\bar{r}\in R} x_{\bar{r}} = \quad m \tag{2.11}$$

$$x_{\bar{r}} \in \{0, 1\} \qquad \bar{r} \in R \tag{2.12}$$

The objective function (2.9) selects a set of the feasible of routes that minimizes the sum of the route costs while equation (2.10) ensures that all customers are served exactly once and equation (2.11) ensures that exactly m vehicles are used. In some variants of the CVRP equation (2.11) is relaxed such that *at most* m vehicles are used or such that there are no restrictions on the number of vehicles used.

A variant of the CVRP that is often studied in the heuristic literature is the distance constrained CVRP, where a distance measure d_{ij} (possibly different from c_{ij}) is assigned to each arc. An upper bound on distance D is also given and no routes must be longer than D. This constraint is easily added to our model, we simply require that the visits $v_0, \ldots, v_{h+1}$ in our feasible path $\bar{r}$ should satisfy

$$\sum_{i=0}^{h} d_{v_i,v_{i+1}} \leq D. \tag{2.13}$$

The constraint (2.13) can also be seen as a limit of time spent on the route and service times at customers can be incorporated in (d_{ij}).

Figure 2.3 shows an optimal solution to a small CVRP instance. Note that routes can cross each other in an optimal solution with euclidean distances. This is caused by the capacity constraint.

The CVRP was introduced by Dantzig and Ramser [1959] and has been subject to intense research since then. Many heuristic methods have been proposed in the last 45 years and it is out of the scope of this section to give an overview of these. Instead we would recommend four surveys. Heuristics proposed up until around 1980 are surveyed in Christofides et al. [1979], while the most successful heuristics until the new millennium are surveyed in Laporte and Semet [2002] and Gendreau et al. [2002]. The most recent advances in metaheuristics have been surveyed in Cordeau et al. [2004]. The best heuristic for the problem at the moment is the metaheuristic proposed by Mester and Bräysy [2005]. The general heuristic, presented in this thesis, is tested on benchmark CVRP instances in Chapter 6. The results show that the heuristic is on par with most of the heuristics proposed for the problem recently, but the heuristic by Mester and Bräysy [2005] produces better results than the heuristic proposed in this thesis for the particular problem.

Quite a lot of attention has been given to exact methods for the CVRP in the recent years and substantial advances in the size of problems that can be solved to optimality has been achieved. Most research has gone into developing branch and cut methods and valid inequalities for the problem. The two most successful branch and cut algorithms are the one proposed by Lysgaard et al. [2004] and Blasum and Hochstättler [2000]. Recently it has been shown that the combination of column generation and cutting planes is a powerful approach for the CVRP and the branch-and-cut-and-price algorithm proposed by Fukasawa et al. [2005] must be considered as the best

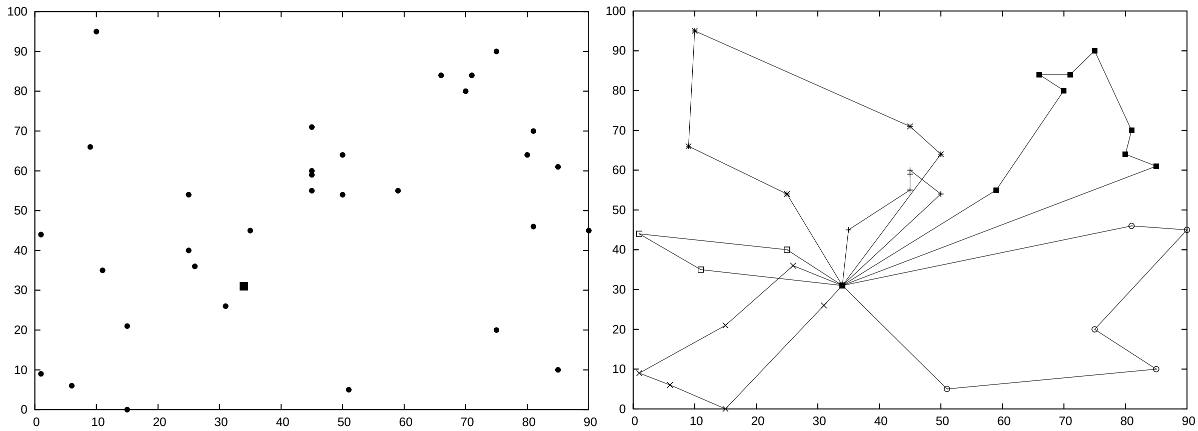


Figure 2.3: CVRP illustration. The figure on the left shows the nodes in a CVRP instance with 32 customers, the figure on the left shows the optimal CVRP solution for this instance.

algorithm currently. This algorithm was able to solve all instances with up to 134 customers that it was tested with. This does not mean that all instances with 130 customers or less can be solved routinely though, as only two instances with more than 100 customers were attempted.

Many more exact methods have been proposed through the years for the CVRP. Most of these are surveyed by Toth and Vigo [2002a], Naddef and Rinaldi [2002], Bramel and Simchi-Levi [2002] and Cordeau et al. [2005b].

2.4 The vehicle routing problem with time windows

The vehicle routing problem with time windows (VRPTW) generalizes the CVRP by associating travel times t_{ij} with arcs (i, j) and service times s_i and time windows $[a_i, b_i]$ with customers i and depot $i = 0$. The vehicle should arrive before or within the time window of a customer. If it arrives before the start of the time window, it has to wait until the time window opens before service at the customer can start. The problem can be modelled using the framework introduced in Section 2.3. To ease the notation, we again consider the depot as split into two nodes. The route $\bar{r} = (v_0, v_1 \ldots, v_h, v_{h+1})$ should satisfy the following criteria in order to be valid. The capacity requirement is identical to the one from equation (2.7):

$$\sum_{i=1}^{h} q_{v_i} \leq Q \tag{2.14}$$

We introduce a variable S_i to indicate when service starts at node i. A route must obey the following constraints to be time feasible

$$a_{v_i} \leq S_{v_i} \leq b_{v_i} \qquad \forall i \in \{0, \ldots, h+1\} \tag{2.15}$$

$$S_{v_{i+1}} \geq S_{v_i} + s_{v_i} + t_{v_i, v_{i+1}} \qquad \forall i \in \{0, \ldots, h\} \tag{2.16}$$

Equation (2.15) ensures that the service start time is within the time window of the node and equation (2.16) updates the start time along the route. The cost of a route is defined as in equation (2.8). Given the set R of feasible VRPTW routes, the VRPTW can now be formulated as

$$\min f(x) \tag{2.17}$$

subject to

$$\sum_{\bar{r} \in R} a_{i\bar{r}} x_{\bar{r}} = 1 \qquad \forall i \in \{1, \ldots, n\} \tag{2.18}$$

$$x_{\bar{r}} \in \{0, 1\} \qquad \bar{r} \in R \tag{2.19}$$

Two different objectives are studied in the literature. The first objective is to minimize the sum of the route costs as in the CVRP, that is $f(x) = \sum_{\bar{r}\in R} c_{\bar{r}} x_{\bar{r}}$. The alternative objective is to minimize the number of vehicles used as first priority and the route costs as second priority. This can be written as $f(x) = M \sum_{\bar{r}\in R} x_{\bar{r}} + \sum_{\bar{r}\in R} c_{\bar{r}} x_{\bar{r}}$, where M is a sufficiently large integer. The first objective function is usually considered in the literature about exact methods for the VRPTW while the second objective is used with heuristics.

Figure 2.4 shows an example of an optimal VRPTW solution. The figure only shows the geometrical aspects, not the time windows. The time windows cause routes to cross themselves, even in optimal solutions.

The amount of heuristics proposed for the VRPTW is exceptional. Especially in the nineties and in the new millennium many metaheuristics have been proposed. A short overview of metaheuristics is given by Cordeau et al. [2002] while a more recent and comprehensive survey is given by Bräysy and Gendreau [2005b]. Another recent survey is presented by Cordeau et al. [2005b]. It is hard to say which metaheuristic that is the best for the VRPTW currently as a heuristic can be judged on many different parameters like speed, robustness and precision. Two good candidates would be the hybrid evolutionary algorithm proposed by Mester and Bräysy [2005] and the general heuristic presented in this thesis. The heuristic proposed in this thesis is particularly well suited for minimizing the number of vehicles necessary to serve all customers, as the computational experiments in Chapter 6 show.

Exact methods for the VRPTW have been surveyed by Cordeau et al. [2002] and Cordeau et al. [2005b]. Exact methods for the VRPTW have been developing rapidly in the recent years. This can be illustrated by the fact that 5 years ago, several instances from the Solomon test set (Solomon [1987]) with 25 customers were still unsolved while today all instances with 25 and 50 customers from the test set have been solved. The last unsolved instances with 50 customers were reported solved this year by Jepsen et al. [2005] and Kallehauge and Boland [2005]. Although neither of the two papers could solve all of the 50 customer problems, the union of the solved instances covers all instances.

It is interesting to note that one of the new inequalities proposed in Kallehauge and Boland [2005], which is one of the reasons for the success of the approach presented in that paper is almost identical to the *fork* inequality proposed in Chapter 8 of this thesis. The two inequalities were developed independently of each other.

Exact solution methods for the VRPTW are dominated by column generation methods, with the branch and cut method by Kallehauge and Boland [2005] as the lone exception. Much of the improvement in exact column generation approaches is due to developments in solving the pricing problem. Prior to Irnich and Villeneuve [2003], Feillet et al. [2004] and Chabrier [2005], the pricing problem that was solved in column generation approaches was the *shortest path problem with time window and capacity constraints* (SPPTWCC) that allowed cycles of length 3 or more in the shortest paths. Irnich and Villeneuve [2003] proposed an algorithm for the pricing problem that eliminated cycles of length k in the shortest paths, where k is a parameter. $k = 2$ corresponds to the traditional pricing problem solved. Irnich and Villeneuve [2003] showed that using $k > 2$ drastically improved the lower bound obtained from the column generation approach. Feillet et al. [2004] and Chabrier [2005] went a step further and solved the *elementary shortest path problem with time window and capacity constraints* (ESPPTWCC) as pricing problem. In the ESPPTWCC the shortest paths have to be simple, that is without any cycles. They empirically showed that the problem is not too hard to solve and that using this pricing problem once more increased the lower bounds to the VRPTW.

Recently Righini and Salani [2004, 2005] proposed improvements to the ESPPTWCC algorithm that resulted in great speed ups. The improvements came from performing a bidirectional search that simultaneously searches for shortest paths from the source and destination nodes and merges the result “when the two searches meet”. Traditional algorithms search from the source node only. Their other contribution is *decremental state space relaxation* that initially solves a SPPTWCC where cycles are allowed and then gradually forbids repetition of the nodes that take part in cycles. This usually improves the running time as the algorithm typically only needs to disallow repetition of a small subset of nodes in order to get an elementary path instead of disallowing

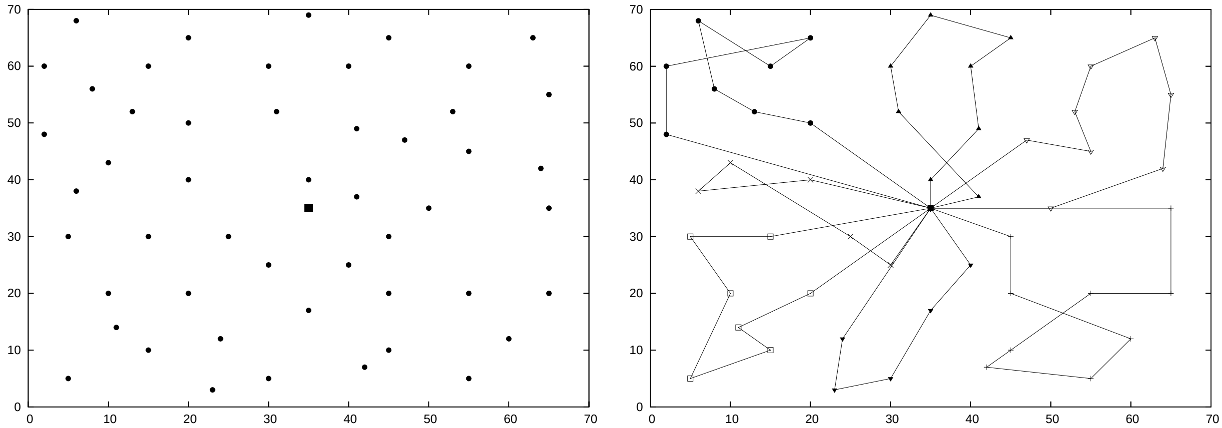


Figure 2.4: VRPTW illustration. The figure on the right shows the optimal solution to instance r107 with 50 nodes from the Solomon data set. The figure on the left shows the nodes in the problem

repetition of all nodes which is more time consuming. It should be noted that the last idea also has been proposed by Boland et al. [2005].

Jepsen et al. [2005] proposed to use valid inequalities from the set-partitioning problem to raise the lower bound obtained by the column generation approach. They examined the *clique inequality* from the set-partitioning problem and showed how the pricing problem must be changed to handle an inequality similar to the clique inequality (a *non-robust* inequality according to the vocabulary introduced in Poggi de Aragão and Uchoa [2003]). The computational results showed that the lower bound was improved significantly when introducing these cuts and several previously unsolved instances in the Solomon set were solved using these inequalities.

The developments that have taken place within column generation for the VRPTW inspired the research into set-partitioning relaxations of the PDPTW, which is presented in Chapter 9.

If the capacity constraint (2.14) is removed from the problem then one gets a *multiple traveling salesman problem with time windows* (m-TSPTW), which is a problem that has received much less attention in the literature compared to its sibling, the VRPTW. A short overview of some papers on the m-TSPTW is given by Cordeau et al. [2002].

2.5 Pickup and delivery problem with time windows

The pickup and delivery problem with time windows (PDPTW) generalizes the VRPTW. In the PDPTW one no longer delivers goods from a depot to the customers, instead the customers need goods to be transported from a *pickup* location to a *delivery* location. Each pickup-delivery pair is called a *request*. The problem is defined on a graph with $2n+2$ nodes, where n is the number of requests. Each request i is associated with node i and $n+i$, where i is the pickup and $n+i$ is the delivery of the request. Node 0 and $2n+1$ represents the terminals where vehicles start (0) and end $(2n+1)$ their trips. A time window $[a_i, b_i]$ is associated with every node in the graph and a load d_i is associated with every node $1 \leq i \leq 2n$. It is assumed that $d_{n+i} = -d_i$ for all $i = 1, \ldots, n$. Just as for the VRPTW, travel times t_{ij} and costs c_{ij} are associated with arcs (i,j) and service times s_i are associated with node i. It is assumed that all vehicles are identical and have capacity Q.

The task in the PDPTW is to construct routes for the vehicles such that the pickup and delivery corresponding to the same request is served by the same vehicle, that the pickup is served before the corresponding delivery and such that time window and the capacity constraints are obeyed. Just as for the VRPTW it is common to either minimize route costs (c_{ij}) or minimize the number of vehicles necessary to serve all requests.

Using the modeling framework from the previous sections, the requirement to a feasible route $\bar{r} = (v_0, v_1 \ldots, v_h, v_{h+1})$ can be stated as follows. The pairing constraint that ensures that the pickup and delivery of a request is served on the same route can be stated as

$$i \in \{v_1, \ldots, v_h\} \Leftrightarrow n + i \in \{v_1, \ldots, v_h\} \qquad \forall i \in \{1, \ldots, n\} \tag{2.20}$$

The precedence constraint between the pickup and delivery node of the same request can be stated as

$$v_j = i \Rightarrow n + i \in \{v_{j+1}, \ldots, v_h\} \qquad \forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, h\} \tag{2.21}$$

The time windows are modeled as for the VRPTW, S_i is a variable that indicates when service starts at node i

$$a_{v_i} \leq S_{v_i} \leq b_{v_i} \qquad \forall i \in \{0, \ldots, h+1\} \tag{2.22}$$
$$S_{v_{i+1}} \geq S_{v_i} + s_{v_i} + t_{v_i, v_{i+1}} \qquad \forall i \in \{0, \ldots, h\} \tag{2.23}$$

Capacity checks are a little more complicated than in the preceding sections as the capacity no longer is increasing monotonously along the route

$$0 \leq \sum_{i=0}^{j} d_{v_i} \leq Q \qquad \forall j \in \{0, \ldots, h+1\} \tag{2.24}$$

As for the VRPTW the typical objectives are to minimize the sum of the arc costs c_{ij} or minimize the number of vehicles used as first priority and then minimize arc costs as second priority.

A more complex variant of the PDPTW is studied in Chapters 4 to 6. This variant includes multiple depots, precedences between nodes not belonging to the same request and site dependencies. Mathematical models for the more complex problem are given in Chapters 4 and 6.

Figure 2.5 shows an optimal solution for a single-depot PDPTW instance. It is clear that the capacity, time windows, pairing and precedence constraints give rise to a quite messy solution.

The literature about the PDPTW is not as extensive as the VRPTW and it is less homogeneous. The PDPTW studied in the literature often contains extra constraints not present in the core formulation presented in this chapter which makes comparison among different methods difficult. In the recent years there has been some tendency in the heuristic community to study the core PDPTW problem though, and a set of common benchmark instances has appeared.

A variant of the PDPTW that has been studied frequently is the *dial-a-ride problem* (DARP). Where the PDPTW usually is thought of as a model for transporting goods, dial-a-ride problems are models for a class of passenger transportation problems. It is frequently used to model the transportation of disabled and elderly people. In this variant of the PDPTW a request consists of transporting one or more persons from one place to another. In contrast to the plain PDPTW, the DARP has constraints or terms in the objective that seek to keep customer inconvenience at a respectable level. How customer inconvenience is modeled differs from paper to paper - there is not one single model that qualifies as *the* model for the DARP. In this thesis a DARP is solved in two of the chapters — Chapters 8 and 9. In the DARP variant considered here, a *max ride time* constraint is enforced on each request. The max ride time constraint ensures that the time from a customer is picked up to the time he is delivered is less than a constant L. Thereby we make sure that no customer is taken on long detours which most likely would annoy the customer even though he makes it to his destination within his time window. This constraint can be expressed as follows in our modeling framework

$$v_i = l \wedge v_j = n + l \Rightarrow S_{v_j} - (S_{v_i} + s_{v_i}) \leq L \qquad \forall i, j \in \{1, \ldots, h\}, \forall l \in \{1, \ldots, n\}. \tag{2.25}$$

Other models for the DARP contains more constraints and penalise user inconvenience in the objective function. Toth and Vigo [1997] for example proposed a model where the customer specifies a pickup time or a delivery time. A time window is constructed around this point in

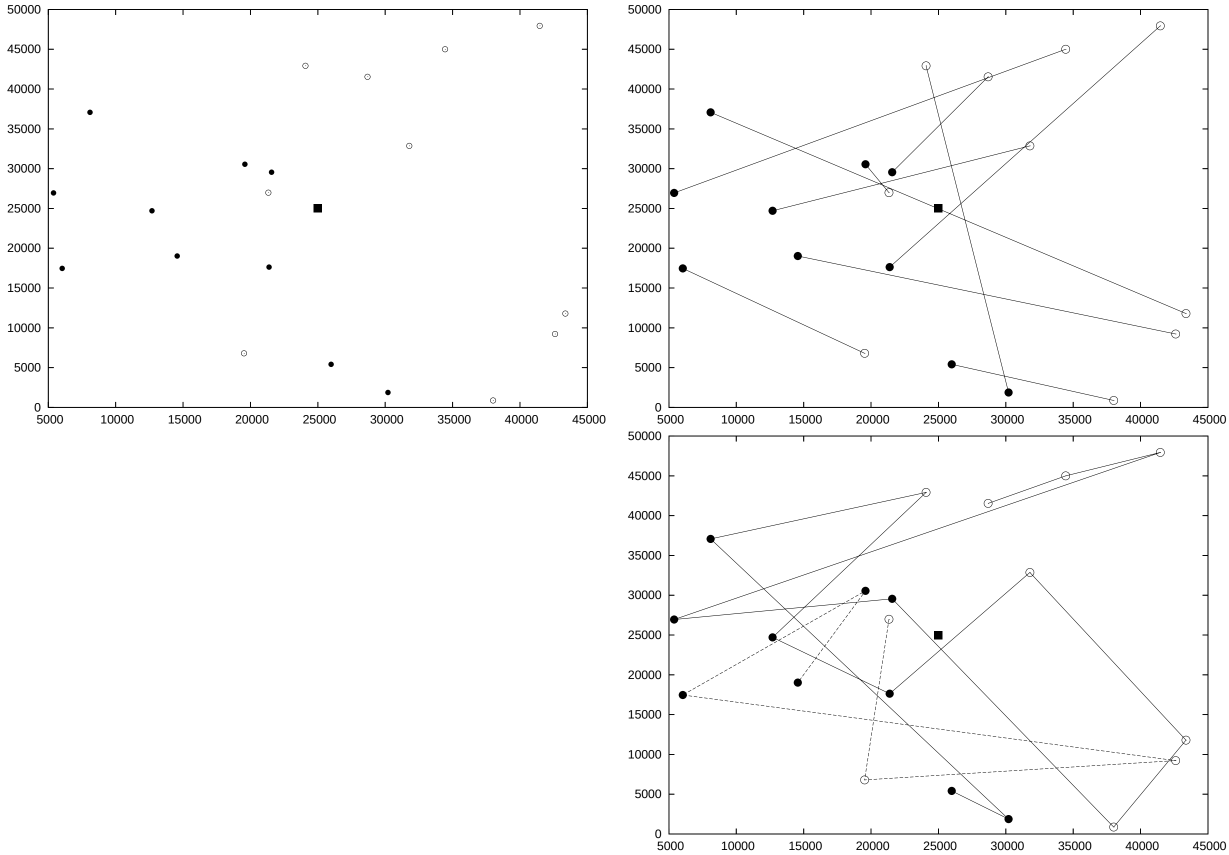


Figure 2.5: PDPTW Figure. Top left: the depot and nodes in the instance. The depot is indicated as a black square, pickups are circles and deliveries are discs. Top right: the requests in the problem, a pickup and a delivery connected by a line forms a request. Bottom right: The optimal PDPTW solution. Two routes are necesarry to serve the requests (shown with solid and dashed lines). Arc between the depot and pickup/delivery nodes are not shown in order to make the figure more readable.

time, and service within the time window is allowed, but a penalty is added to the objective if the vehicle does not arrive at the exact desired time. The model also contains the ride time constraint presented above, but the maximum ride time is dependent on the user. Another feature of the model is that it allows different types of vehicles that have different capabilities. Some vehicles might for example be able to transport a number of wheelchair passengers and some carry trained personnel that can help passengers in need of assistance.

Several surveys of the PDPTW and DARP literature have been presented in the last decade, see Savelsbergh and Sol [1995], Mitrović-Minić [1998], Desaulniers et al. [2002], Cordeau et al. [2005a]. A survey dedicated to the DARP was presented by Cordeau and Laporte [2003].

2.5.1 Heuristics for PDPTW and DARP

Several metaheuristics have been proposed for the PDPTW in recent years and a set of problem instances has appeared as a common platform for testing heuristics. Li and Lim [2001] introduced the set of instances that seems to have become a standard benchmark set for the PDPTW. The instances were constructed from the Solomons test set for the VRPTW (Solomon [1987]) and Gehring and Hombergers larger VRPTW instances (Gehring and Homberger [1999]). The instances were created by first solving the VRPTW instances with a VRPTW heuristic and then pairing nodes that occur in the same route in the VRPTW solution to form a request. The requests are created such that the pickup is visited before the delivery on the VRPTW route. This way of creating PDPTW instances might not result in very realistic instances. One can argue that the requests that are constructed are too "easy" because the pickup and delivery fit well together as they were served by the same route in the VRPTW solution. It turns out that the PDPTW instances created this way are challenging for both heuristics and exact methods, especially the larger instances with 100 requests or more.

The two earliest metaheuristics for the PDPTW were proposed by Gendreau et al. [1998] and Nanry and Barnes [2000]. Gendreau et al. [1998] presented a tabu search for a dynamic version of the PDPTW. They used an interesting neighborhood based on the ejection chain idea: A request i is removed from its route r_1 and reinserted into another route r_2 while ejecting another request j from r_2. j is inserted into a third route, thereby ejecting a third request and so on. The chain of ejections ends with the insertion of the last request into a route without ejecting a new request. Gendreau et al. [1998] describe how a good ejection chain can be found using heuristics.

Nanry and Barnes [2000] used a tabu search algorithm with a neighborhood consisting of three moves: 1) moving a request from one route to another, 2) exchanging a request in one route with a request from another route, and 3) relocating a request to another position within its original route. The heuristic was tested on instances with up to 50 requests. Two other tabu search variants, based on the same neighborhood structure proposed by Nanry and Barnes, were presented a little later by Lau and Liang [2001] and Li and Lim [2001].

Créput et al. [2004] proposed an evolutionary algorithm for the PDPTW where an individual in the solution simply is a solution to the PDPTW. Two crossover methods are proposed, both can produce infeasible offspring, where some requests are not visited or some requests are visited twice. Such offspring are repaired by inserting or removing requests as necessary. The algorithm also incorporates mutation operators based on local search. The heuristic was tested on the 50 request instances from Li and Lim [2001], but the solution quality obtained was worse than that obtained by Li and Lim's tabu search heuristic. Another genetic algorithm was proposed by Pankratz [2005] for the PDPTW where the genetic encoding stores the partitioning of requests on vehicles, but not the actual routing of the requests. The heuristic was tested on instances with around 50 requests from Li and Lim [2001] and Nanry and Barnes [2000]. The computational results seem to be better than the ones obtained by the other genetic algorithm (Créput et al. [2004]). Bent and Hentenryck [2006] applied a two-stage heuristic to the PDPTW and obtained good results on the instances proposed by Li and Lim. The first stage minimizes the number of vehicles used to serve the requests, this is done using a simulated annealing algorithm whose neighborhood consists of moving a request from one position in the solution to another. A modified objective function is used while minimizing the number of vehicles. The modified objective function encourages solutions that contain routes with a few requests and routes with many requests. This objective was chosen from the philosophy that it should be easy to eliminate the short routes. The second stage minimizes the traveled distance using *large neighborhood search* (LNS). The LNS heuristic alternates between removing requests from the current solution and reinserting the requests again. Removal of requests is carried out by a heuristic that removes related requests as proposed by Shaw [1998] for the VRPTW. Re-insertion of the requests is performed using a truncated branch-and-bound search that only allow a certain amount of branching.

Recently Lu and Dessouky [2005] proposed a new insertion algorithm for the PDPTW and tested it on the 50 request instances proposed by Li and Lim [2001]. A non-standard measure called the *crossing length percentage* was taken into account when constructing routes to make the routes more visually attractive. The measure is zero if a route does not cross itself and increases with the number of times it crosses itself, depending on the type of crossing.

Xu et al. [2003] have proposed a heuristic based on column generation to solve a PDPTW inspired by real life cases. The problem considered contains several constraints that have not been studied much in the literature. One of these constraints is that pickup and deliveries must be nested such that the last request loaded is the first one unloaded (LIFO). The model also considers legal working hours of drivers. The heuristic is tested on instances with up to 500 requests and results are looking promising.

Bodin and Sexton [1986] presented a heuristic for a variant of the DARP where customer inconvenience were to be minimized. The heuristic used clustering and local search and were tested on a real-life problem containing 85 requests and 7 vehicles. Jaw et al. [1986] proposed an insertion algorithm for another DARP variant where customers either specify a desired pickup time or a desired delivery time and the heuristic must route the customers such that their pickup or delivery times are sufficiently close to the desired time. The ride time of a customer is furthermore not allowed to surpass a pre-specified acceptable ride time for that customer. The algorithm was tested on a large real-life problem.

Toth and Vigo [1997] presented a heuristic for the variant of the DARP described in Section 2.5. An initial solution is created using a parallel instertion heuristic and this solution is improved upon by using tabu search. The heuristic is tested on real life data and compared to solutions found by human schedulers. This comparison turned out to be difficult to perform as the hand made solutions greatly violated the constraints of the problem. The results indicate that the heuristic did well compared to the hand-made solutions and were fast considering the computer used to perform the experiments.

2.5.2 Exact methods for PDPTW and DARP

Several exact methods for the PDPTW have been proposed in the last 20 years, although the number of papers about this subject is smaller than the amount of literature about the exact solution of the CVRP and VRPTW. There is no established set of benchmark problems used in the exact-PDPTW literature as it is the case in the CVRP and VRPTW community. This makes comparison of different approaches hard, as the hardness of a PDPTW instance depends just as much on its structure as on its size. The paper presented in Chapter 9 tries to improve on this situation by presenting results on the readily available instances proposed by Li and Lim [2001] and on another set of PDPTW instances that are proposed in Chapter 8.

The first exact algorithm for the pure PDPTW was proposed by Desrosiers et al. [1986]. In this paper an exact algorithm for the 1-vehicle PDPTW was described. The algorithm is based on dynamic programming and rules for eliminating dominated labels are defined. The algorithm is able to handle problems with up to 40 requests. In the early nineties a column generation algorithm for the multi vehicle PDPTW was presented by Dumas et al. [1991]. This paper presented clever label domination and label elimination rules and was able to handle instances with up to 50 requests. Later in the nineties Sol [1994] presented another column generation algorithm for the PDPTW. This algorithm differed from the one proposed by Dumas et al. [1991] by using another pricing problem and different branching rules. Sol [1994] also presented new pricing heuristics and procedures for limiting the number of variables in the set partitioning problem. A condensed and updated version of Sol [1994] can be found in Savelsbergh and Sol [1998]. The column generation algorithms presented by Dumas et al. [1991], Sol [1994], Savelsbergh and Sol [1998] form the basis of the column generation algorithms proposed in Chapter 9.

Another column generation algorithm for a variant of the PDPTW was proposed recently by Sigurd et al. [2004]. The application that motivated this study was the transportation of live pigs. Each request corresponds to the transportation of animals from one location to another (e.g. from farm to farm). This application implies that there are extra precedence constraints on the requests to avoid the spread of diseases: a healthy group of pigs must not be transported on a vehicle that previously has transported pigs that have been exposed to some diseases. These precedence rules make it possible to solve the pricing problem on a acyclic, layered graph that allows quick evaluation of the pricing problem for even large instances.

Lübbecke [2001] used column generation to solve an *Engine Scheduling* problem which can be seen as a pickup and delivery problem. The problem was solved with what the author calls *price-and-branch* meaning that columns are generated in the root node only. If the LP relaxation in the root node turns out to be fractional, then a branch and bound search is started, but new columns are not generated in the child nodes in the branch-and-bound tree. This means that the solution found by the price-and-branch approach only is guaranteed optimal if it has the same objective as the lower bound found in the root node. Solutions with a different objective value might be optimal, but there is no guarantee.

Lu and Dessouky [2004] proposed a branch-and-cut algorithm for the PDPTW and the multiple vehicle pickup and delivery problem with capacity constraints (PDP). They presented a compact 2-index model for the problem with a polynomial number of constraints and variables as opposed to the model presented in Chapter 8 of this thesis that contains an exponential number of constraints. Lu and Dessouky presened several valid inequalities to improve the lower bound obtained from the LP relaxation of the model. Problems with up to 25 requests for the PDP and 15 requests for the PDPTW were solved to optimality in the computational experiments. Another branch-and-cut algorithm was proposed for the DARP by Cordeau [2006]. This algorithm forms the basis of the branch-and-cut algorithm proposed in Chapter 8 so we refer to this chapter for further information.

Exact methods for the single vehicle pickup and delivery problem without time window and capacity constraints (PDTSP) have been studied by Kalantari et al. [1985] and Ruland and Rodin [1997]. Kalantari et al. [1985] proposed a branch and bound method using a combinatorial lower bound. Instances with up to 18 requests were solved by this approach. Ruland and Rodin [1997] developed a branch-and-cut algorithm for the undirected version of the problem. The paper introduced new valid inequalities for the problem and instances with up to 15 requests were solved. The valid inequalities presented in this paper were later adapted to the directed case and used in a branch-and-cut algorithm for the dial-a-ride problem by Cordeau [2006]. The model for the basic PDP proposed by Ruland and Rodin [1997] was also used as an inspiration for the model for the PDPTW presented in Chapter 8. Recently Dumitrescu [2005] presented new valid inequalities for the PDTSP and identified classes of facet defining inequalities.

Psaraftis [1980] presented an exact dynamic programming approach for a variant of the single vehicle DARP. In this variant of the DARP, an ordering of the customers is given and in order to minimize customer inconvenience the order the customers are served in must not deviate too much from their initial ordering. An integer *maximum position shift* (MPS) is given and this integer defines how far out of sequence a customer can be picked up or delivered. Furthermore ride time of the customers should be minimized as well as the overall ride time of the vehicle.

Part II

Heuristics

Chapter 3

Introduction to heuristics

3.1 Introduction

This chapter introduces heuristic concepts for vehicle routing problems. The chapter uses the CVRP as the primary example as this is a reasonably simple problem that makes it easy to introduce the necessary concepts.

3.2 Heuristic categories

Heuristics can be categorized broadly into three different categories: *construction heuristics*, *improvement heuristics* and *metaheuristics*. These three categories are explained in the next three sections (Section 3.2.1 to 3.2.4).

Laporte and Semet [2002] proposed a different classification of heuristics for vehicle routing problems. The propose two main classes *classical heuristics* and *metaheuristics*. The class of classical heuristics is divided into three groups: *constructive heuristics*, *two-phase heuristics* and *improvement methods*. The term two-phase heuristics covers heuristics that divide the construction into two phases: a *clustering phase* and a *routing phase*. In the classification of heuristics used in this thesis, two-phase heuristics are seen as construction heuristics.

3.2.1 Construction heuristics

Laporte and Semet [2002] define construction heuristics as follows

Constructive heuristics gradually build a feasible solution while keeping an eye on solution cost, but they do no contain an improvement phase per se.

Many construction heuristics for vehicle routing problems have been proposed during the last 40 years. In the recent years it appears that their popularity has faded somewhat in the scientific literature as metaheuristics have become more dominant, however papers about construction heuristics still appear. Some examples are the PDPTW insertion heuristic by Lu and Dessouky [2005], the VRPTW insertion heuristic by Ioannou et al. [2001] and the savings algorithm for the CVRP by Altinel and Öncan [2005].

Fast heuristics are important from a practical point of view as many real world applications of heuristics require fast response times. In a vehicle routing application one needs to quickly reconstruct part of the solution if an incident happens while carrying out the plan or if a customer calls in with a new transportation task and wants to know if the task can be carried out. Fast construction algorithms are often the preferable algorithm for such situations and for very large problems containing thousands or tens of thousands of customers.

Fast heuristics can also be used as subroutines in more time consuming metaheuristics, this approach is used in this thesis.

Many construction heuristics for vehicle routing problems fall into one of the three classes: *insertion heuristics*, *savings heuristics* and *clustering heuristics*.

Insertion heuristics build a solution by inserting one customer at a time. Insertion heuristics can build one route at a time (*sequential insertion heuristics*) or build many or all routes in parallel (*parallel insertion heuristics*). The choice of which customer to insert and where to insert the customer is what differentiates the insertion heuristics. A very simple insertion heuristic could choose to insert the customer that increases the overall cost the least.

Savings heuristics initially build a solution where each customer is served on its own route. Routes are then merged one by one according to some criteria. Savings algorithms vary by the criterion used for merging routes (what saving is obtained by merging two routes) and by how routes are merged. For the CVRP the most simple merge operation deletes an edge between the depot and a customer from each of the two routes that is being merged and joins the route by adding an edge between the two customers that are adjacent to only one edge. More advanced merging procedures consider all the customers served by the two routes and solve a TSP (in case of the CVRP) on these customers.

The savings heuristic was first proposed by Clarke and Wright [1964] and consequently it is often denoted the *Clarke and Wright algorithm.* Many variants and improvements of the algorithm have been proposed and it has been applied to different variants of vehicle routing problems including a heterogeneous VRPTW (Liu and Shen [1999]) and pickup and delivery problem with full truckloads (Gronalt et al. [2004]), but most savings algorithms have been proposed for the CVRP. New variants of the savings algorithm are still proposed. A recent example is given by Altinel and Öncan [2005].

Clustering algorithms are two-phase algorithms. The first phase consists of grouping customers into subsets (clusters) where each subset should be served by one route. The second phase then creates routes for each subset. A third phase may be employed to repair the solution if it turns out that some of the clusters could not be served by a single vehicle.

Fisher and Jaikumar [1981] presented a clustering heuristic for the CVRP where the number of vehicles is fixed to K. In their approach a number of *seed* customers are selected initially and for each remaining customer i, a heuristic cost d_{ik} of routing customer i with seed customer k is computed. A generalized assignment problem is then solved, using d_{ik} in the objective. This produces K clusters that each satisfies the capacity constraint. Each cluster is turned into a route by solving a TSP to optimality.

Another clustering approach is the sweep algorithm for the CVRP which was presented by Gillet and Miller [1974]. In this algorithm customers are clustered in sectors of the circle around the depot as shown on Figure 3.1. In practice the algorithm works by sorting customers according to their polar coordinate angle with the depot as (0,0). The algorithm starts from the first customer in the list and adds this customer to a cluster. The algorithm continues to process the customers according to the ordering and adds the customer to the current cluster as long as the cluster can be served by a single vehicle. When it is no longer possible to add a customer to the current cluster a new cluster is started and becomes the current cluster. When all customers have been assigned to a cluster a TSP tour is found for each cluster to produce a CVRP solution. Gillet and Miller [1974] also included an improvement phace after the clustering.

3.2.2 Local search heuristics

Local search heuristics are heuristics that take a solution as input, modify this solution by performing a sequence of operations on the solution and produce a new, hopefully improved solution. At all times the heuristic has a *current solution* and it modifies this solution by evaluating the effect of changing the solution in systematic way. If one of the changes leads to an improved solution, then the current solution is replaced by the new improved solution and the process is repeated. In more advanced local search heuristics the algorithm sometimes perform changes that

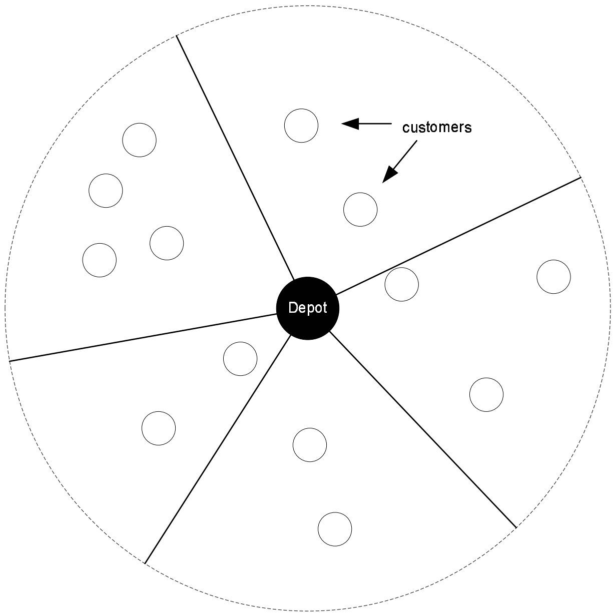


Figure 3.1: Sweep algorithm. Customers in each sector of a circle are served by one vehicle.

lead to a solution that is worse than the current. This is done as one can hope to find an even better solution after a few more changes.

The term *improvement heuristic* [Laporte and Semet [2002]] can be used to describe a local search heuristic that only performs operations that improve the objective of the solution.

In the following we introduce local search heuristics more formally. The presentation follows that of Funke et al. [2005].

We are given an instance I of a combinatorial optimization problem. $\mathcal{S}$ is the set of feasible solutions to the instance and $c : \mathcal{S} \rightarrow \mathbb{Q}$ is a function that maps from a solution to the solution *cost*. $\mathcal{S}$ is assumed to be finite, but it is often an extremely large set as pointed out in Section 1.2.2. We assume that the combinatorial optimization problem is a minimization problem, that is, we want to find the solution s^* for which $c(s^*) \leq c(s) \, \forall s \in \mathcal{S}$.

We define a *neighborhood* of a solution $s \in \mathcal{S}$ as $N(s) \subseteq \mathcal{S}$. That is, N is a function that maps from a solution to a set of solutions. A solution s is said to be *locally optimal* or a *local optimum* with respect to a neighborhood N if $c(s) \leq c(s') \, \forall s' \in N(s)$. With these definitions it is possible to define a steepest descent algorithm (see Algorithm 1). The algorithm takes an initial solution as input (line 1). It repeats line 3–7 as long as it found an improved solution in the last iteration. The neighborhood of s is searched in line 3 and s' is the best solution in the neighborhood. In line 4 it is determined if the new solution is better than the previous. If it is, then we update the current solution in line 5 and reiterate. If the current solution was not improved then the algorithm terminates with the best solution observed during the search. The algorithm is called a steepest descent algorithm as it always chooses the best solution in the neighborhood. Another strategy is to choose the first improving solution observed in the neighborhood. Such an algorithm would be a *descent algorithm*. Funke et al. [2005] use the terms *best search* and *first search* for a steepest descent algorithm and a descent algorithm, respectively.

Another concept in local search heuristics is a *move*. A move m is an operation that transforms a solution s into another, possibly infeasible, solution s' that shares some characteristics of s. Following Funke et al. [2005] we define a superset Z of $\mathcal{S}$ ($\mathcal{S} \subseteq Z$) containing all solutions that can be reached by applying moves to a solution in $\mathcal{S}$. Thus m is a function that maps from Z to Z and M is the set of all moves. The set M defines an extended neighborhood $\hat{N}(s)$ to each solution s, $\hat{N}(s) = \{m(x) : m \in M\}$, $\hat{N}(s) \subseteq Z$ and $N(s) = \hat{N}(s) \cap \mathcal{S}$. The extended neighborhood makes

Algorithm 1 Steepest descent

```
input: Initial solution s ∈ S_I
improved = true
while (improved)
  s' = arg min_{x∈N(s)} {c(x)}
  if c(s') < c(s)
    s = s'
  else
    improved = false
return s
```

it easier to discuss the *size* of a neighborhood, we define the size of a neighborhood as $\left|\hat{N}(s)\right|$ or $|M|$. Using $N(s)$ to measure the size of a neighborhood is problematic as the size of this set would depend on s, but a working definition could be $\max\{|N(s)| : s \in \mathcal{S}\}$.

3.2.3 Neighborhoods

In this section we describe some neighborhoods proposed for vehicle routing problems. It is far from a complete description of all the neighborhoods conceived. Giving such a description is out of the scope of this section. What we wish to convey with this section is an idea of the different kinds of neighborhoods that have been proposed and attempted in practice. For a more complete survey of neighborhoods we refer the reader to Bräysy and Gendreau [2005a] which discusses VRPTW neighborhoods and Funke et al. [2005] for a more general presentation.

VRP neighborhoods can be split into two major categories: *Single-Route Improvements* and *Multiroute Improvements*, following the terminology from Laporte and Semet [2002], or *Single-Route neighborhoods* and *Multiroute neighborhoods* as we prefer to call them. Single-Route neighborhoods perform changes to one route at a time, that is, they permute the customers within a route. Thus TSP neighborhoods can be used as Single-Route neighborhoods for the CVRP; TSPTW neighborhoods can be used for the VRPTW and 1-PDPTW neighborhoods can be used for the PDPTW.

Multiroute neighborhoods exchange and move customers between two or more routes. This implies that they can make greater structural changes to a solution. In the following sections we will only consider multiroute neighborhoods.

3.2.3.1 Small neighborhoods

This section reviews a few classic neighborhoods for vehicle routing problems. The size of the neighborhoods $\left|\hat{N}(s)\right|$ is rather small, that is a small polynomial function of n, the number of customers. The neighborhoods are usually searched explicitly, but tricks to avoid evaluating parts of the neighborhood have also been proposed.

Osman [1993] proposed a quite general neighborhood called the λ-*interchange* that encompasses many of the neighborhoods used in other papers. Given a solution $s = (R_1, \ldots, R_p, \ldots, R_q, \ldots, R_m)$ where R_t are the routes of the solution the λ-interchange selects all pairs of routes (R_p, R_q) and subsets of customers on the routes $S_p \subseteq R_p$ and $S_q \subseteq R_q$ with $|S_p| \leq \lambda$ and $|S_q| \leq \lambda$. The two sets of customers are exchanged and the routes are reoptimized. The λ-interchange neighborhood contains all solutions that can be constructed by selecting customer sets of the given size. The neighborhood quickly grows large and gets difficult to handle when larger lambdas are used. Using $\lambda = 1$ contains the often used *relocate* neighborhood where a move consits of transfering a customer from one route to another and it also contains the *exchange* move that exchanges two customers.

Another class of neighborhoods changes focuses on changing edges in the solution (of course the λ-*interchange* can also be viewed as changing edges, but it's not the object that the neighborhood focuses on). One example is the *2-opt** neighborhood proposed by Potving and Rousseau [1995] for the VRPTW, this neighborhood selects two routes R_p and R_q, deletes an edge in each two route and reconnects the first part of R_p with the last part of R_q and vice versa.

3.2.3.2 Large and exponential sized neighborhoods

This section reviews af few large neighborhoods that has been proposed for vehicle routing problems. Most notably it gives a short introduction to the *Large Neighborhood Search* (LNS) that is used in Chapters 4 to 6.

A precise definition of when a neighborhood is large or small, is not simple to give. One definition could be that a large neighborhood is exponential in the instance size, but that seems a little to restrictive. Ahuja et al. [2002] defines a large neighborhoods the exponential ones and the ones that are too large to search explicitely in practice. We will use this definition.

The LNS heurisitic forms the foundation of the heuristic presented in Chapters 4 to 6. It was first presented as a heuristic framework by Shaw [1998]. The general neighborhood employed can be described in very few words: A move in the LNS consists removing up to q customers and then reinserting these customers into the solution somehow. When implementing the heuristic one has a lot of freedom in determining the rules for chosing the customers to remove and for chosing methods for reinserting them. The remove/reinsert idea has occured before Shaw [1998] formalized it, Russell [1995] for example, proposed a VRPTW improvement heuristic that removes up to 5 customers and reinserts them using partial enumeration. A heuristic similar to LNS idea was also put forward by Schrimpf et al. [2000]. The heuristic proposed recently by Franceschi et al. [2005] can also be characterized as a LNS heuristic although the authors do not make this connection. In this heuristic the customers are reinserted by solving an IP problem to optimality.

The *Adaptive Large Neighborhood Search Heuristic* (ALNS) proposed in Chapters 4 to 6 extends the LNS by not only having one removal methods and one insertion methods, but a whole set of removal/insertion methods, which in practice are fast heuristics. The heuristic to use is selected using an adaptive method that uses statistics from the search so far to make the choice. The computational experiments in this thesis confirms that these two extensions, although simple, improves the performance of the heuristic.

The LNS pricinciple has also been used as a subcomponent in the *AGES* heuristic propoposed by Mester and Bräysy [2005] that currently is the best heuristic for the CVRP and competes with the ALNS heuristic for being the best heuristic for large VRPTW instances. Thus it seems like the neighborhood is very well suited for vehicle routing problems. Several other large neighborhoods have been proposed for vehicle routing problems, but none of them has been as succesful as the LNS.

Another large neighborhood for the VRP is the *cyclic transfers* proposed by Thompson and Orlin [1989]. The cyclic transfer performs a chain of customer relocations: A customer i_1 is moved from its route r_{i_1} to route r_{i_2} where a customer i_2 is removed, this customer is then moved to a new route and so on. At the end of the chain, customer i_p is inserted into route r_{i_1}. If no route is allowed to be repeated on the chain then the problem of finding the best move in the neighborhood can be transformed to a graph problem, the so called *subset disjoint minimum cost cycle problem* (SDMCCP), that unfortunately is NP-hard. So the SDMCCP must in general be solved by heuristics, although Dumitrescu [2002] presents an exact algorithm for the SDMCCP that performs well in some important cases. The cyclic transfer can be extended to moving clusters of customers between routes or to handle chains where no customers are inserted on the route from which the first customer in the chain was taken from. Recently Agarwal et al. [2004] proposed a CVRP heuristic based on the ideas of cyclic transfers that allowed the operations in the chain to be more complex than just relocating a single customer. The heuristic could for example relocate a sequence of customers from one route to another.

The last large neighborhood for VRP we are going to discuss in this section has not received much attention. It was proposed by Hjorring [1995] and is based on the petal method [Ryan et al. [1993]]. The petal method is a construction heuristic proposed for the CVRP. Given an ordering of the customers $i_1, i_2, \ldots, i_n$ the heuristic creates candidates for routes by first considering customer

i_1. For customer i_1 the routes containing customers $\{i_1\}, \{i_1, i_2\}, \{i_1, i_2, i_3\}, \ldots, \{i_1, \ldots, i_p\}$ are created until a customer $i_{p'}$ is met for which the route containing customers $\{i_1, \ldots, i_{p'}\}$ would be infeasible. Then the heuristic goes on to create routes formed by considering customer i_2 and so on. The ordering is cyclic so, for example, the set containing two elements, generated by customer i_n is $\{i_n, i_1\}$. When all the routes have been constructed the optimal selection of routes that serves all customers can be found in polynomial time by solving a series of shortest path problems. Hjorring creates a large CVRP neighborhood out of this procedure by making small pertubations in the ordering of the customers. This might be a small neighborhood in the solution space defined by permutations of customer but it is a large neighborhood in the CVRP solution space as each permutation potentially corresponds to an exponential number of CVRP solutions. A similar idea has later been used by Prins [2004] in a genetic algorithm where each solution in the population is encoded as a permutation of customers.

3.2.4 Metaheuristics

Metaheuristics has been a very popular research area in the last 20 years and very impressive results have been obtained using these heuristics. Several books and survey/tutorial papers have been written about the topic. Consequently, we are not going to present another introduction to metaheuristics, as it would be hard to bring anything new to the field. We assume that the reader is familiar with the topic, if not, the following references are recommended as starting points: Voß [2001], Blum and Roli [2003], Gendreau and Potvin [2005].

The metaheuristic used in this paper is simulated annealing - not so much because it is our favourite metaheuristic but because it seemed easy to integrate with the ALNS. Afterwards we have tried to combine the ALNS with tabu search and iterated local search but we have not been able to obtain a heuristic with the same quality as the original simulated annealing heuristic.

3.3 Trends in heuristic research for the VRP

This section outlines some of the trends in the research in heuristic methods for static vehicle routing problems and it contains some comments on the direction I foresee and/or hope the research will move in the coming years. The section is quite subjective in some paragraphs and other researchers in the VRP community may have different opinions or see different opportunities than I do.

The section first lists some possible research directions and then comments on the impact I believe these directions will have in the future.

- More complex and rich vehicle routing problems.
- Faster heuristics (disregarding increasing computer speeds) that still produces high quality solutions.
- Ability to handle larger instances.
- More precise heuristics - better solution quality without worrying overly about the time needed for the computation.
- Simpler heuristics.
- Heuristics using mathematical programming - combining ideas from exact optimization with heuristics.
- Parallel implementations.
- More realistic test instances.

More complex models. I believe that more complex and rich vehicle routing problems are going to be a subject that will receive significant attention in the near future, and it is a trend that already is present today. It is an important topic as real life problems contain more constraints than what is present in a standard CVRP or VRPTW.

It is a slightly "dangerous" and problematic research path as the result might be many *case studies* papers that apply heuristics to a certain, special problem arising in a given industry, possibly with constraints that are specific to a particular region or political system and not very general. Such studies are of course welcome, but in my opinion it can be hard to distill general knowledge from them and comparison between different models and heuristics can be difficult due to the lack of a common foundation. It is my hope that the research in more complex vehicle route problems is going to continue along the following paths

1. Identify certain structures and constraints occurring in real life problems and transfer these to the scientific community. Introduce the structure or constraint in a clear way that captures the essence of the problem. It is acceptable to leave some detail out of the new model in order to avoid an overly cluttered model.
   An example of this approach is the combination of 2D packing with the vehicle routing problem that recently has been proposed (Iori et al. [2004], Gendreau et al. [2004]). The packing component of the problem is occurring in practice, it has not been considered in the literature before, and it is modeled in a reasonably simple way, such that the model is clear and future researchers can continue working on the problem.
   Note that introducing new constraints, just to introduce a new problem, is not to be recommended. The new constraints should be an interesting contribution in itself.

2. Identify heuristics that are robust and easily adaptable to a variety of problem types. The heuristic presented in Chapters 4 to 6 is an example of one such heuristic. Establishing that a heuristic is robust and adaptable can be done as in this thesis where the heuristic is tested on a number of different problem types, or it can be done by arguing how different problem types could be solved by the heuristic. Another heuristic that has been shown to be easily adaptable to many problem types is the unified tabu search by Cordeau, Laporte and coauthors [Cordeau et al. [1997, 2000], Cordeau and Laporte [2001]].

3. Identify models that are relatively easy to solve by existing heuristics but at the same time are able to express many problem variants. The rich PDPTW used in Chapters 4 to 6 is one such model, but even broader models could be envisioned.

Faster heuristics, larger instances. The quest for faster heuristics has been going on since the beginning of computerized solution of vehicle routing problems, but developments are still taking place and will continue to do so in the future. One of the most important benefits of faster heuristics is that it will allow us to solve larger instances, and this is surely needed in the real world - real world problems are often larger than the 1000 customer instances that typically are the largest instances considered by heuristic methods. Some recent research is worth pointing out, Toth and Vigo [2003] described a way to reduce the running time of tabu search, a method they called *granular tabu search*. The key idea in the granular tabu search is to restrict the neighborhood search by discarding the most unpromising moves. In practice this can be done by looking at the arc lengths and categorize an arc as either promising or unpromising, based on its length but also on other features like if it is incident to the depot or has been used in one of the best solutions encountered so far. When doing the neighborhood search, only moves that involve at least one promising arc are attempted. The approach was tested on CVRP instances with up to around 500 customers and showed that the heuristic was fast considering the computer used.

Another interesting development towards faster heuristics is the *sequential search* for vehicle routing problems, proposed by Irnich et al. [2005]. Sequential search uses techniques developed for local search methods for the TSP to speed up the search of VRP neighborhoods. As opposed to the granular neighborhoods discussed above, sequential search examines the entire neighborhood, but does so implicitly. It is out of the scope of this section to give a complete description of how this is done, but the key idea is that for many standard neighborhoods for vehicle routing problems it is possible to decompose the moves of the neighborhood into so-called *partial moves* that are *cost-independent*. A decomposition is cost-independent if the gain (change in objective function) for the complete move is the sum of the gains of all the partial moves. This is used together with a theorem by Lin and Kernighan [1973] that states that *if a sequence of numbers has a positive sum, there is a cyclic permutation of these numbers such that every partial sum is positive*. This theorem makes it possible to discard many potential moves. The approach is tested on CVRP instances with between 250 to 2500 customers and dramatic improvements over standard implementations of the neighborhood search are obtained. Speedups range from a factor 5 to a factor 800.

This approach is certainly going to be used in future metaheuristics for the CVRP and it will probably be attempted on more complex and constrained problem types like the VRPTW or the PDPTW. It is unknown how powerful the idea is going to be for the more constrained problem types - the computational results in (Irnich et al. [2005]) shows that the speedup decreases when instances become more constrained.

The last contribution toward faster heuristics mentioned here is proposed by Kytöjoki and Bräysy [2005]. They presented a metaheuristic for the CVRP based on variable neighborhood search and guided local search with several implementation tricks to speed up computation. The heuristic was tested on instances with up to 20,000 customers. An instance with 1040 customers could be solved in between 3.4 and 6.6 minutes depending on the heuristic used while the instance with 20000 customers took between 51 and 144 minutes depending on the heuristic. The solution quality seems good.

More precise heuristics. Heuristics that deliver solutions of high quality is a topic that received a lot of attention, especially since the arrival of metaheuristics. I do not think it is as important a research direction any more, as it has once been. It seems like the best of today's heuristics are consistently able to reach solutions whose cost is within 1–1.5% of the optimal or best known solution cost. For many applications of vehicle routing problems this is good enough, as the data that can be collected in real life will be influenced by errors or noise anyway. Consequently, the notion of an optimal solution is not that important when dealing with real life instances in most cases.

Heuristics that produce high quality solutions are nevertheless going to receive attention in the future - one reason is that there always will be a certain personal satisfaction in seeing your heuristic produce solutions better than the previously best known! Another reason is that solution quality is easy to measure and therefore an obvious way of comparing heuristics

Simpler heuristics. Focusing the research toward simpler heuristics was proposed by Gendreau et al. [2002]. The authors write: *It is time to develop simpler methods capable of quickly providing good quality solutions*. I certainly agree that a simple heuristic is preferable (by far) to a complicated one, but I do not feel that this is the way the research in general is moving and has been moving in the recent years. Occasionally we will see simple heuristics appear, and we will learn from those, perhaps more than from the complicated heuristics that are able to improve upon best known solutions. But I believe that the ideas from these simpler heuristics are going to be combined with other ideas to form more and more complicated heuristics due to the competitive nature of the field.

It is worthwhile to consider if the ALNS heuristic proposed in this thesis is a simple heuristic. I believe that the basic idea in the heuristic is simple and can be described in 1 page. The description of the heuristic gets complicated if the sub-heuristics that define its neighborhoods have to be explained, and the implementation of the heuristic itself is complicated. We have gone to some lengths to try not to make the heuristic overly complicated. For example, we have avoided trying to incorporate local searches based on more traditional neighborhoods even though this could have improved the results somewhat.

Mathematical programming based heuristics. A line of research that I believe is going to be studied more in the future is a combination of ideas from heuristics with exact optimization and mathematical programming. The best heuristics in terms of solution quality for the two most famous vehicle routing problems, the CVRP and VRPTW, typically contain very few applications of theoretical results. These heuristics are usually based on a clever exploration of the neighborhood, that is, a good trade-off between intensification and diversification, and algorithmic techniques to speed up the evaluation of the neighborhood. Thus it seems likely that a more mathematic approach could provide some new insights and improvements. Heuristics based on mathematical models have been proposed recently, some examples are heuristic column generation [Xu et al. [2003], Sigurd et al. [2005]], neighborhood evaluation by transformation to a graph problem [Agarwal et al. [2004], Ergun et al. [2002]], neighborhood search through a polynomial solvable set-partitioning problem [Hjorring [1995]], clustering based on a Lagrangian lower bound [Toth and Vigo [1999]] and a removal and reinsertion based approach where insertion is done by solving a set-partitioning problem [Franceschi et al. [2005]]. The last is actually able to find some very high quality solutions to the CVRP if given a very good initial solution. The computation time is very large (A problem with 120 customers took more than a day to solve on a modern PC), but it nevertheless shows that there is some hope for using mathematical models within heuristics.

Parallel heuristics. The current trend in CPU architectures is that improvements in clock frequencies are beginning to stagnate and chip makers are placing multiple cores in their CPUs in order to improve performance. The top-level workstation CPUs from AMD and Intel today have two cores on the CPU and CPUs with even 4 or even 8 cores are on Intel's road map. In a few years single core CPUs might become obsolete. In order to get the full performance from these multi-core CPUs one needs to consider parallel programming. It is going to be interesting to see how big an impact this development is going to have on the heuristic community. A recent book about parallel metaheuristic is [Alba [2005]].

More realistic test instances. I hope that more test instances from the industry will become available to the scientific community. Most of the instances we test our heuristics on are generated by some random process, and it is uncertain how well these instances mimic real life instances. Unfortunately it is often hard to release real life instances to the public. Many companies, from which the data originates, considers such data as confidential. A step toward more realistic instances could be to generate data in a more clever manner. For example to get a more realistic geographic distribution of customers, one could look up the addresses of persons with a certain, common last name in a specified area and record their addresses. These addresses could be turned into coordinates (the process is known as *geocoding*). This would produce a geographic distribution that mimics that found in a delivery problem to private customers - customers would be clustered in urban areas. To make instances even more realistic, road network distances could be used instead of Euclidean distances — this should have a significant impact in an area like Denmark where there are many islands and fjords in certain parts of the country.

The lack of realistic instances is perhaps most evident when looking at the large scale instances used to compare heuristics for the CVRP. One set of instances for the CVRP contains 20 instances with the number of customers ranging between 240 and 483 has been proposed by Golden et al. [1998] and is accepted in the literature as a standard set for large CVRP problems. Another set, containing 12 instances with up to 1200 customers was proposed recently by Li et al. [2005]. This set has not been used much in the literature yet, but it will likely be used more in the future. The only papers I am aware of that use the instances are Li et al. [2005], Kytöjoki and Bräysy [2005] and Chapter 6 of this thesis.

All of these instances are highly symmetrical, an example from the second set is shown in Figure 3.2. The instances were created this way to make it easy to establish a good solution by hand, and this solution can be compared to the heuristic solution, but in my opinion it is problematic that all of the large scale instances that we test our CVRP heuristics on have this property. The instances, certainly do not look like the instances occurring in real life and we risk creating a generation of heuristics that are particular well suited at solving these symmetrical problems, but that might be less robust toward more general customer configurations. It is therefore my hope that another data set will appear for the CVRP and be used on equal terms with the existing data sets. One candidate for such a date set could be the one used by Irnich et al. [2005].

The unified heuristic presented in this thesis is only tested on the instances by Golden et al. [1998] and Li et al. [2005] as well as a classic data set by Christofides et al. [1979]. The instances by Irnich et al. [2005] were unknown to us at the time when the paper in Chapter 6 was submitted.

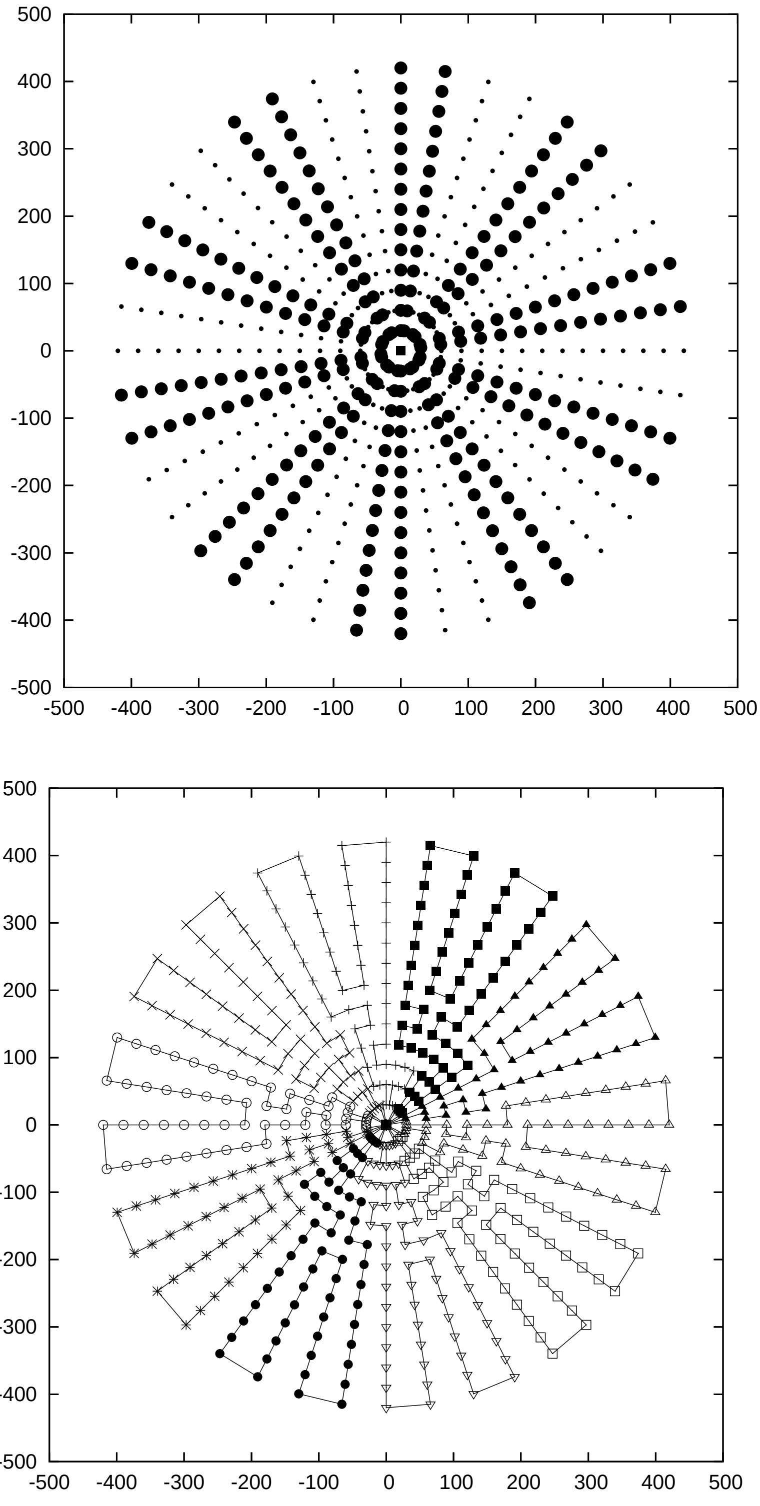


Figure 3.2: Large CVRP instance, 560 customers (instance named "21" in Li et al. [2005]) . The top figure shows the customers in the problem. Large circles have demand 30, small circles have demand 10. The capacity of the vehicles is 1200. The bottom figure shows a solution found by the ALNS heuristic with cost 16224.81, the different point styles mark different routes. The best known solution has cost 16212.74.

3.3.1 Trends in heuristic research for the VRP - conclusion

The preceding section has outlined a number of research areas within the area of heuristics for vehicle routing problems that I believe are going to receive attention in the coming years.

With the enormous amount of literature on heuristics for the vehicle routing problem, a natural question is: *is there really much left to do*? My answer to that question is “yes”. I believe that researchers will continue to be challenged to make even better, more general and robust heuristics for vehicle routing problems in the next decade, just as they have been in the last decade.

Designing and implementing heuristics for vehicle routing problems is a very popular topic in the operations research community, which not necessarily only is a good thing. The reasons for the popularity are probably the obvious applicability of the problem and the low barrier for entering the field: the problems are easy to understand, the benchmark instances are easy to obtain and the standard heuristics do not require much theoretical insight to understand. These are also some of the reasons why I entered the field.

The low barrier for entering implies that many heuristics are proposed - some of them have a quality that I believe is below what is acceptable. The many heuristics also creates a field that is hard to get an overview of - for example, I believe that only a few researchers in the community have thorough knowledge of all the heuristics that have been proposed for the VRPTW through the last 15 years.

I hope that the heuristic papers in this thesis show that it is not necessary to propose a new heuristic for every combination of the classic constraints that one can think of. It certainly is possible to design a heuristic that can handle a variety of combinations and still produce good results.

Chapter 4

An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows

An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows

Stefan Ropke, David Pisinger

Abstract

The *pickup and delivery problem with time windows* is the problem of serving a number of transportation *requests* using a limited amount of vehicles. Each request involves moving a number of goods from a pickup location to a delivery location. Our task is to construct routes that visit all locations such that corresponding pickups and deliveries are placed on the same route and such that a pickup is performed before the corresponding delivery. The routes must also satisfy time window and capacity constraints.

This paper presents a heuristic for the problem based on an extension of the *Large Neighborhood Search* heuristic previously suggested for solving the vehicle routing problem with time windows. The proposed heuristic is composed of a number of competing sub-heuristics which are used with a frequency corresponding to their historic performance. This general framework is denoted *Adaptive Large Neighborhood Search*.

The heuristic is tested on more than 350 benchmark instances with up to 500 requests. It is able to improve the best known solutions from the literature for more than 50% of the problems.

The computational experiments indicate that it is advantageous to use several competing sub-heuristics instead of just one. We believe that the proposed heuristic is very robust and is able to adapt to various instance characteristics.

Keywords: Pickup and Delivery Problem with Time Windows, Large Neighborhood Search, Simulated Annealing, Metaheuristics

Introduction

In the considered variant of the pickup and delivery problem with time windows (PDPTW), we are given a number of *requests* and *vehicles*. A request consists of picking up goods at one location and delivering these goods to another location. Two time windows are assigned to each request: a pickup time window that specifies when the goods can be picked up and a delivery time window that tells when the goods can be dropped off. Furthermore *service times* are associated with each pickup and delivery. The service times indicate how long it will take for the pickup or delivery to be performed. A vehicle is allowed to arrive at a location before the start of the time window of the location, but the vehicle must then wait until the start of the time window before initiating the operation. A vehicle may never arrive to a location after the end of the time window of the location.

Each request is assigned a set of feasible vehicles. This can for example be used to model situations where some vehicles cannot enter a certain location because of the dimensions of the vehicle.

Each vehicle have a limited capacity and it starts and ends its duty at given locations called *start* and *end terminals*. The start and end location do not need to be the same and two vehicles can have different start and end terminals. Furthermore each vehicle is assigned a start and end time. The start time indicates when the vehicle must leave its start location and the end time denotes the latest allowable arrival at its end location. Note that the vehicle leaves its depot at the specified start time even though this may introduce a waiting time at the first location visited.

Our task is to construct valid routes for the vehicles. A route is valid if time windows and capacity constraints are obeyed along the route, each pickup is served before the corresponding delivery, corresponding pickup and deliveries are served on the same route and the vehicle only serves requests it is allowed to serve. The routes should be constructed such that they minimize the *cost* function to be described below.

As the number of vehicles is limited, we might encounter situations where some requests cannot be assigned to a vehicle. These requests are placed in a virtual *request bank*. In a real world situation it is up to a human operator to decide what to do with such requests. The operator might for example decide to rent extra vehicles in order to serve the remaining requests.

The objective of the problem is to minimize a weighted sum consisting of the following three components: 1) the sum of the distance traveled by the vehicles, 2) the sum of the time spent by each vehicle. The time spent by a vehicle is defined as its arrival time at the end terminal minus its start time (which is given a priori), 3) the number of requests in the request bank. The three terms are weighted by the coefficients α, β and γ respectively. Normally a high value is assigned to γ in order to serve as many requests as possible. A mathematical model is presented in section 1 to define the problem precisely.

The problem was inspired from a real life vehicle routing problem related to transportation of raw materials and goods between production facilities of a major Danish food manufacturer. For confidentiality reasons, we are not able to present any data about the real life problem that motivated this research.

The problem is NP-hard as it contains the traveling salesman problem as a special case. The objective of this paper is to develop a method for finding good but not necessarily optimal solutions to the problem described above. The developed method should preferably be reasonably fast, robust and able to handle large problems. Thus it seems fair to turn to heuristic methods.

The next paragraphs survey recent work on the PDPTW. Although none of the references mentioned below consider exactly the same problem as ours, they all face the same core problem.

Nanry and Barnes [15] are among the first to present a metaheuristic for the PDPTW. Their approach is based on a Reactive Tabu Search algorithm that combines several standard neighborhoods. In order to test the heuristic, Nanry and Barnes create PDPTW instances from a set of standard VRPTW problems proposed by Solomon [26]. The heuristic is tested on instances with up to 50 requests. Li and Lim [11] use a hybrid metaheuristic to solve the problem. The heuristic combines Simulated Annealing and Tabu search. Their method is tested on the 9 largest instances from Nanry and Barnes [15] and they consider 56 new instances based on Solomon's VRPTW problems [26]. Lim, Lim and Rodrigues [12] apply "Squeaky wheel" optimization and local search to the PDPTW. Their heuristic is tested on the set of problems proposed by Li and Lim [11]. Lau and Liang [10] also apply Tabu search to PDPTW and they describe several construction heuristics for the problem. Special attention is given to how test problems can be constructed from VRPTW instances.

Recently, Bent and Van Hentenryck [2] proposed a heuristic for the PDPTW based on Large Neighborhood Search. The heuristic was tested on the problems proposed by Li and Lim [11]. The heuristic by Bent and Van Hentenryck is probably the most promising metaheuristic for the PDPTW proposed so far.

Gendreau et al. [9] consider a dynamic version of the problem. An ejection chain neighborhood is proposed and steepest descent and Tabu search heuristics based on the ejection chain neighborhood are tested. The tabu search is parallelized and the sequential and parallelized versions are compared.

Several column generation methods for PDPTW have been proposed. These methods both include exact and heuristic methods. Dumas et al. [8] were the first to use column generation for solving PDPTW. They propose a branch and bound method that is able to handle problems with up to 55 requests.

Xu et al. [29] consider a PDPTW with several extra real-life constraints, including multiple time windows, compatibility constraints and maximum driving time restrictions. The problem is solved using a column generation heuristic. The paper considers problem instances with up to 500 requests.

Sigurd et al. [24] solve a PDPTW problem related to transportation of livestock. This introduces some extra constraints, such as precedence relations among the requests, meaning that some requests must be served before others in order to avoid the spread of diseases. The problem is solved to optimality using column generation. The largest problems solved contain more than 200 requests.

A recent survey of pickup and delivery problem literature was made by Desaulniers et al. [7].

The work presented in this paper is based on the Masters Thesis of Ropke [19]. In the papers by Pisinger and Ropke [16], [20] it is shown how the heuristic presented in this paper can be extended to solve a variety of vehicle routing problems, for example the VRPTW, the *Multi Depot Vehicle Routing Problem* and the *Vehicle Routing Problem with Backhauls*.

The rest of this paper is organized as follows: Section 1 define the PDPTW problem formally, Section 2 describes the basic solution method in a general context; Section 3 describes how the solution method has been applied to PDPTW and extensions to the method are presented; Section 4 contains the results of the computational tests. The computational test is focused on comparing the heuristic to existing metaheuristics and evaluating if the refinements presented in Section 3 improve the heuristic; Section 5 concludes the paper.

1 Mathematical model

This section presents a mathematical model of the problem, it is based on the model proposed by Desaulniers et al. [7]. The mathematical model serves as a formal description of the problem. As we solve the problem heuristically we do not attempt to write the model on integer-linear form.

A problem instance of the pickup and delivery problem contains n requests and m vehicles. The problem is defined on a graph, $P = \{1,\cdots,n\}$ is the set of pickup nodes, $D = \{n+1,\cdots,2n\}$ is the set of delivery nodes. Request i is represented by nodes i and $i+n$. K is the set of all vehicles, $|K| = m$. One vehicle might not be able to serve all requests, as an example a request might require that the vehicle has a freezing compartment. K_i is the set of vehicles that are able to serve request i and $P_k \subseteq P$ and $D_k \subseteq D$ are the set of pickups and deliveries, respectively, that can be served by vehicle k, thus for all i and k: $k \in K_i \Leftrightarrow i \in P_k \wedge i \in D_k$. Requests where $K_i \neq K$ are called *special requests*. Define $N = P \cup D$ and $N_k = P_k \cup D_k$. Let $\tau_k = 2n+k$, $k \in K$ and $\tau'_k = 2n+m+k$, $k \in K$ be the nodes that represents the start and end terminal, respectively, of vehicle k. The graph $G = (V,A)$ consists of the nodes $V = N \cup \{\tau_1,\cdots,\tau_m\} \cup \{\tau'_1,\cdots,\tau'_m\}$ and the arcs $A = V \times V$. For each vehicle we have a subgraph $G_k = (V_k, A_k)$, where $V_k = N_k \cup \{\tau_k\} \cup \{\tau'_k\}$ and $A_k = V_k \times V_k$. For each edge $(i,j) \in A$ we assign a distance $d_{ij} \geq 0$ and a travel time $t_{ij} \geq 0$. It is assumed that distances and times are nonnegative; $d_{ij} \geq 0, t_{ij} \geq 0$ and that the times satisfy the triangle inequality; $t_{ij} \leq t_{il} + t_{lj}$ for all $i,j,l \in V$. For the sake of modeling we also assume that $t_{i,n+i} + s_i > 0$, this makes elimination of sub tours and the pickup-before-delivery constraint easy to model.

Each node $i \in V$ has a service time s_i and a time window $[a_i, b_i]$. The service time represents the time needed for loading and unloading and the time window indicates when the visit at the particular location must start; a visit to node i can only take place between time a_i and b_i. A vehicle is allowed to arrive to a location before the start of the time window but it has to wait until the start of the time window before the visit can be performed. For each node $i \in N$, l_i is the amount of goods that must be loaded onto the vehicle at the particular node, $l_i \geq 0$ for $i \in P$ and $l_i = -l_{i-n}$ for $i \in D$. The capacity of vehicle $k \in K$ is denoted C_k.

Four types of decision variables are used in the mathematical model. x_{ijk}, $i,j \in V, k \in K$ is a binary variable which is one if the edge between node i and node j is used by vehicle k and zero otherwise. S_{ik}, $i \in V, k \in K$ is a nonnegative integer that indicates when vehicle k starts the service at location i, L_{ik}, $i \in V, k \in K$ is a nonnegative integer that is an upper bound on the amount of goods on vehicle k after servicing node i. S_{ik} and L_{ik} are only well-defined when vehicle k actually visits node i. Finally z_i, $i \in P$ is a binary variable that indicates if request i is placed in the request bank. The variable is one if the request is placed in the request bank and zero otherwise.

A mathematical model is:

$$\min \alpha \sum_{k \in K} \sum_{(i,j) \in A} d_{ij} x_{ijk} + \beta \sum_{k \in K} \left(S_{\tau'_k,k} - a_{\tau_k} \right) + \gamma \sum_{i \in P} z_i \tag{1}$$

Subject to:

$$\sum_{k\in K_i}\sum_{j\in N_k} x_{ijk} + z_i = 1 \qquad \forall i \in P \tag{2}$$

$$\sum_{j\in V_k} x_{ijk} - \sum_{j\in V_k} x_{j,n+i,k} = 0 \qquad \forall k \in K, \forall i \in P_k \tag{3}$$

$$\sum_{j\in P_k\cup\{\tau'_k\}} x_{\tau_k,j,k} = 1 \qquad \forall k \in K \tag{4}$$

$$\sum_{i\in D_k\cup\{\tau_k\}} x_{i,\tau'_k,k} = 1 \qquad \forall k \in K \tag{5}$$

$$\sum_{i\in V_k} x_{ijk} - \sum_{i\in V_k} x_{jik} = 0 \qquad \forall k \in K, \forall j \in N_k \tag{6}$$

$$x_{ijk} = 1 \Rightarrow S_{ik} + s_i + t_{ij} \le S_{jk} \qquad \forall k \in K, \forall (i,j) \in A_k \tag{7}$$

$$a_i \le S_{ik} \le b_i \qquad \forall k \in K, \forall i \in V_k \tag{8}$$

$$S_{ik} \le S_{n+i,k} \qquad \forall k \in K, \forall i \in P_k \tag{9}$$

$$x_{ijk} = 1 \Rightarrow L_{ik} + l_j \le L_{jk} \qquad \forall k \in K, \forall (i,j) \in A_k \tag{10}$$

$$L_{ik} \le C_k \qquad \forall k \in K, \forall i \in V_k \tag{11}$$

$$L_{\tau_k k} = L_{\tau'_k k} = 0 \qquad \forall k \in K \tag{12}$$

$$x_{ijk} \in \{0,1\} \qquad \forall k \in K, \forall (i,j) \in A_k \tag{13}$$

$$z_i \in \{0,1\} \qquad \forall i \in P \tag{14}$$

$$S_{ik} \ge 0 \qquad \forall k \in K, \forall i \in V_k \tag{15}$$

$$L_{ik} \ge 0 \qquad \forall k \in K, \forall i \in V_k \tag{16}$$

The objective function minimizes the weighted sum of the distance traveled, the sum of the time spent by each vehicle, and the number of requests not scheduled.

Constraint (2) ensures that each pickup location is visited or that the corresponding request is placed in the request bank. Constraint (3) ensures that the delivery location is visited if the pickup location is visited and that the visit is performed by the same vehicle. Constraints (4) and (5) ensure that a vehicle leaves every start terminal and a vehicle enters every end terminal. Together with constraint (6) this ensures that consecutive paths between τ_k and τ'_k are formed for each $k \in K$.

Constraints (7), (8) ensure that S_{ik} is set correctly along the paths and that the time windows are obeyed. These constraints also make sub tours impossible. Constraint (9) ensures that each pickup occur before the corresponding delivery. Constraints (10),(11) and (12) ensure that the load variable is set correctly along the paths and that the capacity constraints of the vehicles are respected.

2 Solution method

Local search heuristics are often built on neighborhood moves that make small changes to the current solution, such as moving a request from one route to another or exchanging two requests as in Nanry and Barnes [15] and Li and Lim [11]. These kind of local search heuristics are able to investigate a huge number of solutions in a short time, but a solution is only changed very little in each iteration. It is our belief that such heuristics can have difficulties in moving from one promising area of the solution space to another, when faced with tightly constrained problems, even when embedded in metaheuristics.

One way of tackling this problem is by allowing the search to visit infeasible solutions by relaxing some constraints; see e.g. Cordeau et al. [5]. We take another approach — instead of using small “standard moves” we use very large moves that potentially can rearrange up to 30-40% of all requests in a single iteration. The price of doing this is that the computation time needed for performing and evaluating the moves becomes much larger compared to the smaller moves. The number of solutions evaluated by the proposed heuristic per time unit is only a fraction of the solutions that could be evaluated by a standard heuristic. Nevertheless very good performance is observed in the computational tests as demonstrated in Section 4.

The proposed heuristic is based on *Large Neighborhood Search (LNS)* introduced by Shaw [21]. The LNS heuristic has been applied to the VRPTW with good results (see Shaw[21], [22] and Bent and Van Hentenryck

Algorithm 1 LNS heuristic

```
 1 Function LNS(s ∈ {solutions}, q ∈ ℕ )
 2   solution s_best=s;
 3   repeat
 4     s′ = s;
 5     remove q requests from s′
 6     reinsert removed requests into s′;
 7     if (f(s′) < f(s_best)) then
 8       s_best = s′;
 9     if accept(s′, s) then
10       s = s′;
11   until stop-criterion met
12  return s_best;
```

[4]). Recently the heuristic has been applied to the PDPTW as well (Bent and Van Hentenryck [2]). The LNS heuristic itself is similar to the *ruin and recreate* heuristic proposed by Schrimpf et al. [23].

The pseudo-code for a minimizing LNS heuristic is shown in Algorithm 1. The pseudo-code assumes that an initial solution s already has been found, for example by a simple construction heuristic. The second parameter q determines the scope of the search.

Lines 5 and 6 in the algorithm are the interesting part of the heuristic. In line 5, a number of requests are removed from the current solution s' and in line 6 the requests are reinserted into the current solution again. The performance and robustness of the overall heuristic is very dependent on the choice of removal and insertion procedures. In the previously proposed LNS heuristics for VRPTW or PDPTW (see for example Shaw [21] or Bent and Van Hentenryck [2]) near-optimal methods were used for the reinsert operation. This was achieved using a truncated branch and bound search. In this paper we take a different approach by using simple insertion heuristics for performing the insertions. Even though the insertion heuristics themselves usually deliver solutions of poor quality, the quality of the LNS heuristic is very good as the bad moves that are generated by the insertion heuristics lead to a fruitful diversification of the search process.

The rest of the code updates the so far best solution and determines if the new solution should be accepted. A simple accept criteria would be to accept all improving solutions. Such a criteria has been used in earlier LNS implementations (Shaw [21]). In this paper we use a simulated annealing accept criteria.

In line 11 we check if a stop criterion is met. In our implementation we stop when a certain number of iterations has been performed.

The parameter $q \in \{0, \cdots, n\}$ determines the size of the neighborhood. If q is equal to zero then no search at all will take place as no requests are removed. On the other hand if q is equal to n then the problem is resolved from scratch in each iteration. In general, one can say that the larger q is, the easier it is to move around in the solution space, but when q gets larger each application of the insertion procedure is going to be slower. Furthermore if one uses a heuristic for inserting requests, then choosing q too large might give bad results.

The LNS local search can be seen as an example of a very large scale neighborhood search as presented by Ahuja et al. in [1]. Ahuja et al. define very large scale neighborhoods as neighborhoods whose sizes grow exponentially as a function of the problem size, or neighborhoods that simply are too large to be searched explicitly in practice. The LNS local search fits into the last category, as we have a large number of possibilities for choosing the requests to remove and a large number of possible insertions. One important difference between the proposed heuristic and most of the heuristics described in [1] is that the latter heuristics typically examine a huge number of solutions, albeit implicitly, while the LNS heuristic proposed in this paper only examines a relatively low number of solutions.

Instead of viewing the LNS process as a sequence of remove-insert operations, it can also be viewed as a sequence of *fix-optimize* operations. In the fix operation a number of elements in the current solution are fixed. If for example the solution is represented as a vector of variables, the fix operation could fix a number of these variables at their current value. The optimize operation then re-optimizes the solution while respecting the fixation performed in the previous fix-operation. This way of viewing the heuristic might help us to apply the heuristic to problems where the remove-insert operations do not seem intuitive. In Section 3 we introduce the term *Adaptive Large Neighborhood Search* (ALNS) to describe an algorithm using several large neighborhoods in an adaptive way. A more general presentation of the ALNS framework can be found in the subsequent paper [16].

3 LNS applied to PDPTW

This section describes how the LNS heuristic has been applied to the PDPTW. Compared to the LNS heuristic developed for the VRPTW and PDPTW by Shaw [21], [22] and Bent and Van Hentenryck [2], [4] the heuristic in this paper is different in several ways:

1. We are using several removal and insertion heuristics during the same search while the earlier LNS heuristics only used one method for removal and one method for insertions. The removal heuristics are described in Section 3.1 and the insertion heuristics are described in Section 3.2. The method for selecting which sub-heuristic to use is described in Section 3.3. The selection mechanism is guided by statistics gathered during the search, as described in Section 3.4. We are going to use the term *Adaptive Large Neighborhood Search* (ALNS) heuristic for a LNS heuristic that uses several competing removal and insertion heuristics and chooses between using statistics gathered during the search.
2. Simple and fast heuristics are used for the insertion of requests as opposed to the more complicated branch and bound methods proposed by Shaw [21], [22] and Bent and Van Hentenryck [2], [4].
3. The search is embedded in a simulated annealing metaheuristic where the earlier LNS heuristics used a simple descent approach. This is described in Section 3.5.

The present section also describes how the LNS heuristic can be used in a simple algorithm designed for minimizing the number of vehicles used to serve all requests. The vehicle minimization algorithm only works for homogeneous fleets without an upper bound on the number of vehicles available.

3.1 Request removal

This section describes three removal heuristics. All three heuristics take a solution and an integer q as input. The output of the heuristic is a solution where q requests have been removed. The heuristics *Shaw removal* and *Worst removal* furthermore have a parameter p that determines the degree of randomization in the heuristic.

3.1.1 Shaw removal heuristic

This removal heuristic was proposed by Shaw in [21, 22]. In this section it is slightly modified to suit the PDPTW. The general idea is to remove requests that are somewhat similar, as we expect it to be reasonably easy to shuffle similar requests around and thereby create new, perhaps better solutions. If we choose to remove requests that are very different from each other then we might not gain anything when reinserting the requests as we might only be able to insert the requests at their original positions or in some bad positions. We define the similarity of two requests i and j using a *relatedness measure* $R(i,j)$. The lower $R(i,j)$ is, the more related are the two requests.

The relatedness measure used in this paper consists of four terms: a distance term, a time term, a capacity term and a term that considers the vehicles that can be used to serve the two requests. These terms are weighted using the weights φ, χ, ψ and ω respectively. The relatedness measure is given by:

$$\begin{aligned} R(i,j) \quad = \quad & \varphi\left(d_{A(i),A(j)} + d_{B(i),B(j)}\right) + \chi\left(\left|T_{A(i)} - T_{A(j)}\right| + \left|T_{B(i)} - T_{B(j)}\right|\right) \\ & + \psi\left|l_i - l_j\right| + \omega\left(1 - \frac{\left|K_i \bigcap K_j\right|}{\min\left\{\left|K_i\right|, \left|K_j\right|\right\}}\right) \end{aligned} \tag{17}$$

$A(i)$ and $B(i)$ denote the pickup and delivery locations of request i and T_i indicates the time when location i is visited. d_{ij}, l_i and K_i are defined in Section 1. Using the decision variable S_{ik} from Section 1, we can write T_i as $T_i = \sum_{k \in K} \sum_{j \in V_k} S_{ik} x_{ijk}$. The term weighted by φ measures distance, the term weighted by χ measures temporal

Algorithm 2 Shaw Removal

1 **Function** ShawRemoval($s \in \{solutions\}$, $q \in \mathbb{N}$, $p \in \mathbb{R}_+$)
2 request : r = a randomly selected request from S;
3 set of requests : $D = \{r\}$;
4 **while** $|D| < q$ **do**
5 r = a randomly selected request from D;
6 Array : L = an array containing all request from s not in D;
7 sort L such that $i < j \Rightarrow R(r, L[i]) < R(r, L[j])$;
8 choose a random number y from the interval[0,1);
9 $D = D \bigcup \{L[y^p |L|]\}$;
10 **end while**
11 remove the requests in D from s;

Algorithm 3 Worst Removal

Function WorstRemoval($s \in \{solutions\}$, $q \in \mathbb{N}$, $p \in \mathbb{R}_+$)
while q > 0 **do**
Array : L = All planned requests i, sorted by descending $cost(i,s)$;
choose a random number y in the interval[0,1);
request : r = $L[y^p |L|]$;
remove r from solution s;
$q = q - 1$;
end while

connectedness, the term weighted by ψ compares capacity demand of the requests and the term weighted by ω ensures that two requests get a high relatedness measure if only a few or no vehicles are able to serve both requests. It is assumed that d_{ij}, T_x and l_i are normalized such that $0 \leq R(i,j) \leq 2(\varphi + \chi) + \psi + \omega$. This is done by scaling d_{ij}, T_x and l_i such that they only take on values from $[0,1]$. Notice that we cannot calculate $R(i,j)$, if request i or j is placed in the request bank.

The relatedness is used to remove requests in the same way as described by Shaw [21]. The procedure for removing requests is shown in pseudo code in Algorithm 2. The procedure initially chooses a random request to remove and in the subsequent iterations it chooses requests that are similar to the already removed requests. A determinism parameter $p \geq 1$ introduces some randomness in the selection of the requests (a low value of p corresponds to much randomness).

Notice that the sorting in line 7 can be avoided in an actual implementation of the algorithm, as it is sufficient to use a linear time selection algorithm [6] in line 9.

3.1.2 Random removal

The random removal algorithm simply selects q requests at random and removes them from the solution. The random removal heuristic can be seen as a special case of the Shaw removal heuristic with $p = 1$. We have implemented a separate random removal heuristic though, as it obviously can be implemented to run faster than the Shaw removal heuristic.

3.1.3 Worst removal

Given a request i served by some vehicle in a solution s we define the *cost* of the request as $cost(i,s) = f(s) - f_{-i}(s)$ where $f_{-i}(s)$ is the cost of the solution without request i (the request is not moved to the request bank, but removed completely). It seems reasonable to try to remove requests with high cost and inserting them at another place in the solution to obtain a better solution value, therefore we propose a removal heuristic that removes requests with high $cost(i,s)$.

The worst removal heuristic is shown in pseudo-code in Algorithm 3. It reuses some of the ideas from Section 3.1.1.

Notice that the removal is randomized, with the degree of randomization controlled by the parameter p like in Section 3.1.1. This is done to avoid situations where the same requests are removed over and over again.

One can say that the Shaw removal heuristic and the worst removal heuristic belong to two different classes of removal heuristics. The Shaw heuristic is biased towards selecting requests that “easily” can be exchanged, while the worst-removal selects the requests that appear to be placed in the wrong position in the solution.

3.2 Inserting requests

Insertion heuristics for vehicle routing problems are typically divided into two categories: *sequential* and *parallel* insertion heuristics. The difference between the two classes is that sequential heuristics build one route at a time while parallel heuristics construct several routes at the same time. Parallel and sequential insertion heuristics are discussed in further detail in [17]. The heuristics presented in this paper are all parallel. The reader should observe that the insertion heuristic proposed here will be used in a setting where they are given a number of partial routes and a number of requests to insert — they seldom build the solution from scratch.

3.2.1 Basic greedy heuristic

The basic greedy heuristic is a simple construction heuristic. It performs at most n iterations as it inserts one request in each iteration. Let $\Delta f_{i,k}$ denote the change in objective value incurred by inserting request i into route k at the position that increases the objective value the least. If we cannot insert request i in route k, then we set $\Delta f_{i,k} = \infty$. We then define c_i as $c_i = \min_{k \in K} \{\Delta f_{i,k}\}$. In other words, c_i is the “cost” of inserting request i at its best position overall. We denote this position by *the minimum cost position*. Finally we choose the request i that minimizes

$$\underset{i \in U}{min} \quad c_i \tag{18}$$

and insert it at its minimum cost position. U is the set of unplanned requests. This process continues until all requests have been inserted or no more requests can be inserted.

Observe that in each iteration we only change one route (the one we inserted into), and we do not have to recalculate insertion costs in all the other routes. This property is used in the concrete implementation to speed up the insertion heuristics.

An obvious problem with this heuristic is that it often postpones the placement of “hard” requests (requests which are expensive to insert, that is requests with large c_i) to the last iterations where we do not have many opportunities for inserting the requests as many of the routes are “full”. The heuristic presented in the next section tries to circumvent this problem.

3.2.2 Regret heuristics

The *regret* heuristic tries to improve upon the basic greedy heuristic by incorporating a kind of look ahead information when selecting the request to insert. Let $x_{ik} \in \{1, \ldots, m\}$ be a variable that indicates the route for which request i has the k'th lowest insertion cost, that is $\Delta f_{i,x_{ik}} \leq \Delta f_{i,x_{ik'}}$ for $k \leq k'$. Using this notation we can express c_i from Section 3.2.1 as $c_i = \Delta f_{i,x_{i1}}$. In the regret heuristic we define a *regret value* c_i^* as $c_i^* = \Delta f_{i,x_{i2}} - \Delta f_{i,x_{i1}}$. In other words, the regret value is the difference in the cost of inserting the request in its best route and its second best route. In each iteration the regret heuristic chooses to insert the request i that maximizes

$$\underset{i \in U}{max} \quad c_i^*$$

The request is inserted at its minimum cost position. Ties are broken by selecting the insertion with lowest cost. Informally speaking, we choose the insertion that we will regret most if it is not done now.

The heuristic can be extended in a natural way to define a class of regret heuristics: the *regret-k* heuristic is the construction heuristic that in each construction step chooses to insert the request i that maximizes:

$$\underset{i \in U}{max} \quad \left\{ \sum_{j=1}^{k} \left(\Delta f_{i,x_{ij}} - \Delta f_{i,x_{i1}} \right) \right\} \tag{19}$$

If some requests cannot be inserted in at least $m-k+1$ routes, then the request that can be inserted in the fewest number of routes (but still can be inserted in at least one route) is inserted. Ties are broken by selecting the request with best insertion cost. The request is inserted at its minimum cost position. The regret heuristic presented at the start of this section is a regret-2 heuristic and the basic insertion heuristic from Section 3.2.1 is a regret-1 heuristic because of the tie-breaking rules. Informally speaking, heuristics with $k > 2$ investigate the cost of inserting a request on the k best routes and insert the request whose cost difference between inserting it into the best route and the $k-1$ best routes is largest. Compared to a regret-2 heuristic, regret heuristics with large values of k discover earlier that the possibilities for inserting a request become limited.

Regret heuristics have been used by Potvin and Rousseau [17] for the VRPTW. The heuristic in their paper can be categorized as a regret-k heuristic with $k = m$, as all routes are considered in an expression similar to (19). The authors do not use the change in the objective value for evaluating the cost of an insertion, but use a special cost function. Regret heuristics can also be used for combinatorial optimization problems outside the vehicle routing domain, an example of an application to the Generalized Assignment Problem was described by Martello and Toth [13].

As in the previous section we use the fact that we only change one route in each iteration to speed up the regret heuristic.

3.3 Choosing a removal and an insertion heuristic

In Section 3.1 we defined three removal heuristics (`shaw`, `random` and `worst` removal), and in Section 3.2 we defined a class of insertion heuristics (`basic insertion`, `regret-2`, `regret-3`, etc.). One could select one removal and one insertion heuristic and use these throughout the search, but in this paper we propose to use all heuristics. The reason for doing this is that for example the regret-2 heuristic may be well suited for one type of instance while the regret-4 heuristic may be the best suited heuristic for another type of instance. We believe that alternating between the different removal and insertion heuristics gives us a more robust heuristic overall.

In order to select the heuristic to use, we assign weights to the different heuristics and use a *roulette wheel selection principle*. If we have k heuristics with weights w_i, $i \in \{1,2,\cdots,k\}$, we select heuristic j with probability

$$\frac{w_j}{\sum_{i=1}^{k} w_i} \tag{20}$$

Notice that the insertion heuristic is selected independently of the removal heuristic (and vice versa). It is possible to set these weights by hand, but it can be a quite involved process if many removal and insertion heuristics are used. Instead an adaptive weight adjusting algorithm is proposed in Section 3.4.

3.4 Adaptive weight adjustment

This section describes how the weights w_j introduced in Section 3.3 can be automatically adjusted using statistics from earlier iterations.

The basic idea is to keep track of a score for each heuristic, which measures how well the heuristic has performed recently. A high score corresponds to a successful heuristic. The entire search is divided into a number of *segments*. A segment is a number of iterations of the ALNS heuristic; here we define a segment as 100 iterations. The score of all heuristics is set to zero at the start of each segment. The score of a heuristic is increased by either σ_1, σ_2 or σ_3 in the following situations:

Parameter	Description
σ_1	The last remove-insert operation resulted in a new global best solution.
σ_2	The last remove-insert operation resulted in a solution that has not been accepted before. The cost of the new solution is better than the cost of current solution.
σ_3	The last remove-insert operation resulted in a solution that has not been accepted before. The cost of the new solution is worse than the cost of current solution, but the solution was accepted.

The case for σ_1 is clear: if a heuristic is able to find a new overall best solution, then it has done well. Similarly if a heuristic has been able to find a solution that has not been visited before and it is accepted by the accept criteria in the ALNS search then the heuristic has been successful as it has brought the search forward. It seems sensible to distinguish between the two situations corresponding to parameters σ_2 and σ_3 because we prefer heuristics that can improve the solution, but we are also interested in heuristics that can diversify the search and these are rewarded by σ_3. It is important to note that we only reward unvisited solutions. This is to encourage heuristics that are able to explore new parts of the solution space. We keep track of visited solutions by assigning a hash key to each solution and storing the key in a hash table.

In each iteration we apply two heuristics: a removal heuristic and an insertion heuristic. The scores for both heuristics are updated by the same amount as we can not tell whether it was the removal or the insertion that was the reason for the "success".

At the end of each segment we calculate new weights using the recorded scores. Let w_{ij} be the weight of heuristic i used in segment j as the weight used in formula (20). In the first segment we weight all heuristics equally. After we have finished segment j we calculate the weight for all heuristics i to be used in segment $j+1$ as follows:

$$w_{i,j+1} = w_{ij}(1-r) + r\frac{\pi_i}{\theta_i}$$

π_i is the score of heuristic i obtained during the last segment and θ_i is the number of times we have attempted to use heuristic i during the last segment. The *reaction factor* r controls how quickly the weight adjustment algorithm reacts to changes in the effectiveness of the heuristics. If r is zero then we do not use the scores at all and stick to the initial weights. If r is set to one then we let the score obtained in the last segment decide the weight.

Figure 1 shows an example of how the weights of the three removal heuristics progress over time for a certain problem instance. The plots are decreasing because of the simulated annealing acceptance criteria to be described in the next section. Towards the end of the search we only accept good moves and therefore it is harder for the heuristic to get high scores.

3.5 Acceptance and stopping criteria

As described in Section 2 a simple acceptance criteria would be to only accept solutions that are better than the current solution. This would give us a descent heuristic like the one proposed by Shaw [21]. However, such a heuristic has a tendency to get trapped in a local minimum so it seems sensible to, on occasion, accept solutions that are worse than the current solution. To do this, we use the acceptance criteria from simulated annealing. That is, we accept a solution s' given the current solution s with probability $e^{-\frac{f(s')-f(s)}{T}}$ where $T > 0$ is the *temperature*.

The temperature starts out at T_{start} and is decreased every iteration using the expression $T = T \cdot c$, where $0 < c < 1$ is the *cooling rate*. A good choice of T_{start} is dependent on the problem instance at hand, so instead of specifying T_{start} as a parameter we calculate T_{start} by inspecting our initial solution. First we calculate the cost z' of this solution using a modified objective function. In the modified objective function, γ (cost of having requests in the request bank) is set to zero. The start temperature is now set such that a solution that is w percent worse than the current solution is accepted with probability 0.5. The reason for setting γ to zero is that this parameter typically is large and could cause us to set the starting temperature to a too large number if the initial solution had some requests in the request bank. Now w is a parameter that has to be set. We denote this parameter the *start temperature control parameter*.

The algorithm stops when a specified number of LNS iterations have passed.

3.6 Applying noise to the objective function

As the proposed insertion heuristics are quite myopic, we believe that it is worthwhile to randomize the insertion heuristics such that they do not always make the move that seems best locally. This is achieved by adding a noise term to the objective function. Every time we calculate the cost C of an insertion of a request into a route, we also calculate a random number *noise* in the interval $[-maxN, maxN]$ and calculate the modified insertion costs $C' = max\{0, C + noise\}$. At each iteration we decide if we should use C or C' to determine the insertions

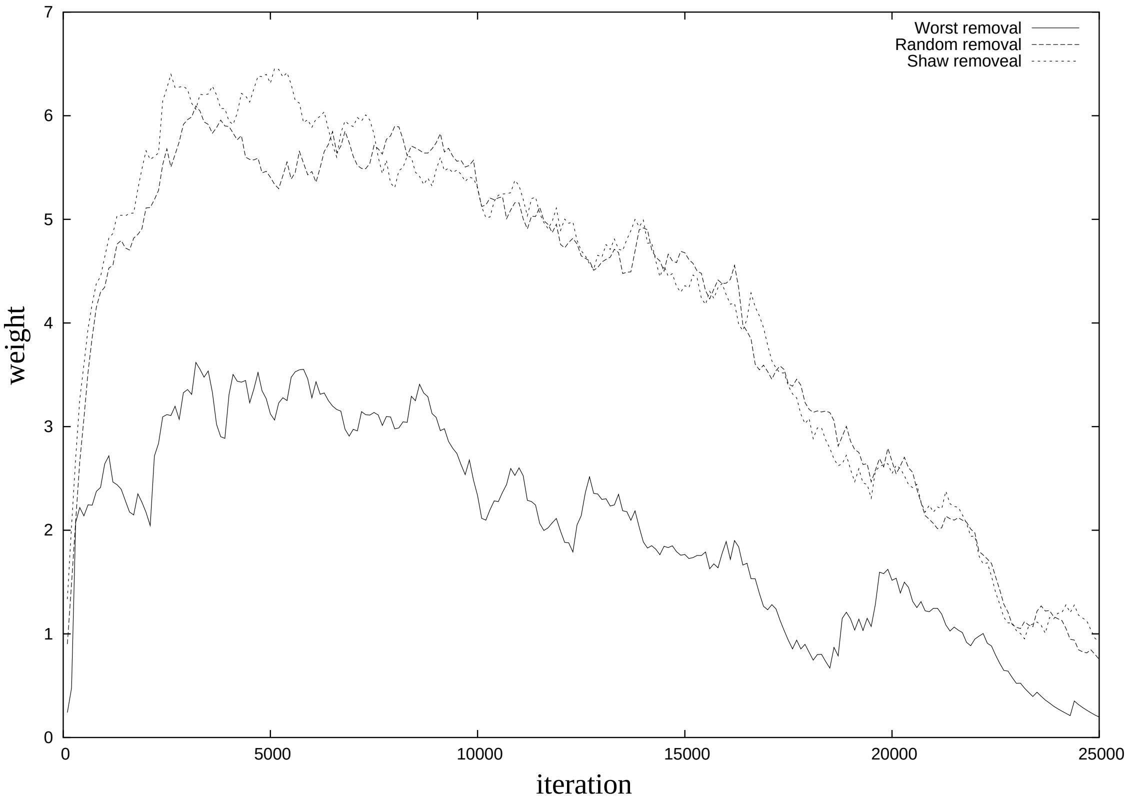


Figure 1: The figure shows an example of how the weights for the three removal heuristics progressed during one application of the heuristic. The iteration number is shown along the x-axis and the weight is shown along the y-axis. The graph illustrates that for the particular problem, the *random* removal and the *Shaw* removal heuristics perform virtually equally well, while the *worst* heuristic performs worst. Consequently the *worst* heuristic is not used as often as the two other heuristics.

to perform. This decision is taken by the adaptive mechanism described earlier by keeping track of how often the noise applied insertions and the "clean" insertions are successful.

In order to make the amount of noise related to the properties of the problem instance, we calculate $maxN = \eta \cdot \max_{i,j \in V} \left\{ d_{ij} \right\}$, where η is a parameter that controls the amount of noise. We have chosen to let $maxN$ be dependent on the distances d_{ij} as the distances are an important part of the objective in all of the problems we consider in this paper.

It might seem superfluous to add noise to the insertion heuristics as the heuristics are used in a simulated annealing framework that already contains randomization, however we believe that the noise applications are important as our neighborhood is searched by means of the insertion heuristics and not randomly sampled. Without the noise applications we do not get the full benefit of the simulated annealing metaheuristic. This conjecture is supported by the computational experiments reported in table 3.

3.7 Minimizing the number of vehicles used

Minimization of the number of vehicles used to serve all requests is often considered as first priority in the vehicle routing literature. The heuristic proposed so far is not able to cope with such an objective, but by using a simple two stage algorithm that minimizes the number of vehicles in the first stage and then minimizes a secondary objective (typically traveled distance) in the second stage, we can handle such problems. The vehicle minimization algorithm only works for problems with a homogeneous fleet. We also assume that the number of vehicles available is unlimited, such that constructing an initial feasible solution always can be done.

A two-stage method was also used by Bent and Van Hentenryck [4], [2], but while they used two different neighborhoods and metaheuristics for the two stages, we use the same heuristic in both stages.

The vehicle minimization stage works as follows: first an initial feasible solution is created using a sequential insertion method that constructs one route at a time until all requests have been planned. The number of vehicles used in this solution is the initial estimate on the number of vehicles necessary. Next step is to remove one route from our feasible solution. The requests on the removed route are placed in the request bank. The resulting problem is solved by our LNS heuristic. When the heuristic is run, a high value is assigned to γ such that requests are moved out of the request bank if possible. If the heuristic is able to find a solution that serves all requests, a new candidate for the minimum number of vehicles has been found. When such a solution has been found, the LNS heuristic is immediately stopped, one more route is removed from the solution and the process is reiterated. If the LNS heuristic terminates without finding a solution where all requests are served, then the algorithm steps back to the last solution encountered in which all requests were served. This solution is used as a starting solution in the second stage of the algorithm, which simply consists of applying the normal LNS heuristic.

In order to keep the running time of the vehicle minimization stage down, this stage is only allowed to spend Φ LNS iterations all together such that if the first application of the LNS heuristic for example spends a iterations to find a solution where all requests are planned, then the vehicle minimization stage is only allowed to perform $\Phi - a$ LNS iterations to minimize the number of vehicles further. Another way to keep the running time limited is to stop the LNS heuristic when it seems unlikely that a solution exists in which all requests are planned. In practice this is implemented by stopping the LNS heuristic if 5 or more requests are unplanned and no improvement in the number of unplanned requests has been found in the last τ LNS iterations. In the computational experiments Φ was set to 25000 and τ was set to 2000.

3.8 Discussion

Using several removal and insertion heuristics during the search may be seen as using local search with several neighborhoods. To the best of our knowledge this idea has not been used in the LNS literature before. The related Variable Neighborhood Search (VNS) was proposed by Mladenović and Hansen [14]. VNS is a metaheuristic framework using a parameterized family of neighborhoods. The metaheuristic has received quite a lot of attention in the recent years and has provided impressive results for many problems. Where ALNS makes use of several unrelated neighborhoods, VNS typically is based on a single neighborhood which is searched with variable depth.

Several metaheuristics can be used at the top level of ALNS to help the heuristic escape a local minimum. We have chosen to use simulated annealing as the ALNS heuristic already contains the random sampling element. For a further discussion of metaheuristic frameworks used in connection with ALNS see the subsequent paper [16].

The request bank is an entity that makes sense for many real life applications. In the problems considered in Section 4 we do not accept solutions with unscheduled requests, but the request bank allows us to visit infeasible solutions in a transition stage, improving the overall search. The request bank is particularly important when minimizing the number of vehicles.

4 Computational experiments

In this section we describe our computational experiments. We first introduce a set of tuning instances in Section 4.1. In Section 4.2 we evaluate the performance of the proposed construction heuristics on the tuning instances. In Section 4.3 we describe how the parameters of the ALNS heuristic were tuned, and in Section 4.4 we present the results obtained by the ALNS heuristic and a simpler LNS heuristics.

4.1 Tuning instances

First a set of representative tuning instances is identified. The tuning instances must have a fairly limited size as we want to perform numerous experiments on the tuning problems and they should somehow be related to the problems our heuristic is targeted at. In the case at hand we want to solve some standard benchmark instances and a new set of randomly generated instances.

Our tuning set consists of 16 instances. The first four instances are LR1_2_1, LR202, LRC1_2_3, and LRC204 from Li and Lim's benchmark problems [11], containing between 50 and 100 requests. The number of available vehicles was set to one more than that reported by Li and Lim to make it easier for the heuristic to find solutions with no requests in the request bank. The last 12 instances are randomly generated instances.

These instances contain both single depot and multi depot problems and problems with requests that only can be served by a subset of the vehicle fleet. All randomly generated problems contain 50 requests.

4.2 Evaluation of construction heuristics

First we examine how the simple construction heuristics from Section 3.2 perform on the tuning problems, to see how well they work without the LNS framework. The construction heuristics regret-1, regret-2, regret-3, regret-4 and regret-*m* have been implemented. Table 1 shows the results of the test. As the construction heuristics are deterministic, the results were produced by applying the heuristics to each of the 16 test problems once.

	Basic greedy	Regret-2	Regret-3	Regret-4	Regret-*m*
Avg. gap (%)	40.7	30.3	26.3	26.0	27.7
Fails	3	3	3	2	0
Time (s)	0.02	0.02	0.02	0.02	0.03

Table 1: Performance of construction heuristics. Each column in the table corresponds to one of the construction heuristics. These simple heuristics were not always able to construct a solution where all requests are served, hence for each heuristic we report the number of times this happened in the *fails* row. The *Avg. gap* row shows the average relative difference between the found solution and the best known solution. Only solutions where all requests are served are included in the calculations of the average relative difference. The last row shows the average time (in seconds) needed for applying the heuristic to one problem, running on a 1.5 GHz Pentium IV.

The results show that the proposed construction heuristics are very fast, but also very imprecise. Basic greedy is the worst heuristic, while all the regret heuristics are comparable with respect to the solution quality. Regret-*m* stands out though, as it is able to serve all requests in all problems. It would probably be possible to improve the results shown in Table 1 by introducing seed requests as proposed by e.g. Solomon [26]. However we are not going to report on such experiments in this paper. It might be surprising that these very imprecise heuristics can be used as the foundation of a much more precise local search heuristic, but as we are going to see in the following sections, this is indeed possible.

4.3 Parameter tuning

This part of the paper serves two purposes. First it describes how the parameters used for producing the results in Section 4.4 were found. Next, it tries to unveil which part of the heuristic contributes most to the solution quality.

4.3.1 Parameters

This section determines the parameters that need to be tuned. We first review the removal parameters. Shaw removal is controlled by five parameters: φ, χ, ψ, ω and p, while the worst removal is controlled by one parameter p_{worst}. Random removal has no parameters. The insertion heuristics are parameter free when we have chosen the regret degree.

In order to control the acceptance criteria we use two parameters, w and c. The weight adjustment algorithm is controlled by four parameters, σ_1, σ_2, σ_3 and r. Finally we have to determine a noise rate η and a parameter ξ that controls how many requests we remove in each iteration. In each iteration, we chose a random number ρ that satisfies $4 \leq \rho \leq \min(100, \xi n)$, and remove ρ requests.

We stop the search after 25000 LNS iterations as this resulted in a fair trade-off between time and quality.

4.3.2 LNS parameter tuning

Despite the large number of parameters used in the LNS heuristic, it turns out that it is relatively easy to find a set of parameters that works well for a large range of problems. We use the following strategy for tuning the parameters: first a fair parameter setting is produced by an ad-hoc trial-and-error phase, this parameter setting was found while developing the heuristic. This parameter setting is improved in the second phase by allowing one parameter to take a number of values, while the rest of the parameters are kept fixed. For each parameter setting we apply the heuristic on our set of test problems five times, and the setting that shows the best average behavior (in terms of average deviation from the best known solutions) is chosen. We now move on to the next parameter, using the values found so far and the values from the initial tuning for the parameters that have not been considered yet. This process continues until all parameters have been tuned. Although it would be possible to process the parameters once again using the new set of parameters as a starting point to further optimize the parameters, we stopped after one pass.

One of the experiments performed during the parameter tuning sought to determine the value of the parameter ξ that controls how many requests we remove and insert in each iteration. This parameter should intuitively have a significant impact on the results our heuristic is able to produce. We tested the heuristic with ξ ranging from 0.05 to 0.5 with a step size of 0.05. Table 2 shows the influence of ξ. When ξ is too low the heuristic is not able to move very far in each iteration, and it has a higher chance of being trapped in one suboptimal area of the search space. On the other hand, if ξ is large then we can easily move around in the search space, but we are stretching the capabilities of our insertion heuristics. The insertion heuristics work fairly well when they must insert a limited number of requests into a partial solution, but they cannot build a good solution from scratch as seen in Section 4.2. The results in Table 2 shows that $\xi = 0.4$ is a good choice. One must notice that the heuristic gets slower when ξ increases because the removals and insertions take longer when more requests are involved, thus the comparison in Table 2 is not completely fair.

ξ	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
Avg. gap (%)	1.75	1.65	1.21	0.97	0.81	0.71	0.81	0.49	0.57	0.57

Table 2: Parameter ξ vs. solution quality. The first row shows the values of the parameter ξ that were tested and the second row shows the gap between the average solution obtained and the best solutions produced in the experiment.

The complete parameter tuning resulted in the following parameter vector $(\varphi, \chi, \psi, \omega, p, p_{worst}, w, c, \sigma_1, \sigma_2, \sigma_3, r, \eta, \xi)$ = (9, 3, 2, 5, 6, 3, 0.05, 0.99975, 33, 9, 13, 0.1, 0.025, 0.4). Our experiments also indicated that it was possible to improve the performance of the vehicle minimization algorithm by setting (w, c) = (0.35, 0.9999) while searching for solutions that serve all requests. This corresponds to a higher start temperature and a slower cooling rate. This indicates that more diversification is needed when trying to minimize the number of vehicles, compared to the situation where one just minimizes the traveled distance.

In order to tune the parameters we start from an initial guess, and then tune one parameter at a time. When all parameters are tuned, the process is repeated. In this way the calibration order plays a minor order. Although the parameter tuning is quite time consuming, it could easily be automated. In our subsequent papers [20, 16] where 11 variants of the vehicle routing problem are solved using the heuristic proposed in this paper we only re-tuned a few parameters and obtained very convincing results, so it seems that a complete tuning of the parameters only needs to be done once.

4.3.3 LNS configurations

This section evaluates how the different removal and insertion heuristics behave when used in a LNS heuristic. In most of the test cases a simple LNS heuristic was used that only involved one removal heuristic and one insertion heuristic. Table 3 shows a summary of this experiment.

The first six experiments aim at determining the influence of the removal heuristic. We see that Shaw removal performs best, the worst removal heuristic is second, and the random removal heuristic gives the worst performance. This is reassuring as it shows that the two slightly more complicated removal heuristics actually are better than the simplest removal heuristic. These results also illustrate that the removal heuristic can have a rather large impact on the solution quality obtained, thus experimenting with other removal heuristics would be interesting and could prove beneficial.

The next eight experiments show the performance of the insertion heuristics. Here we have chosen Shaw removal as removal heuristic because it performed best in the previous experiments. In these experiments we see that all insertion heuristics perform quite well, and they are quite hard to distinguish from each other. Regret-3 and Regret-4 coupled with noise addition are slightly better than the rest though. An observation that applies to all experiments is that application of noise seems to help the heuristic. It is interesting to note that the

	Conf.	Shaw	Rand	Worst	Reg-1	Reg-2	Reg-3	Reg-4	Reg-*m*	Noise	Avg. gap (%)
	1			•		•					2.7
	2			•		•				•	2.6
	3		•			•					5.4
	4		•			•				•	3.2
	5	•				•					2.0
	6	•				•				•	1.6
LNS	7	•			•						2.2
	8	•			•					•	1.6
	9	•					•				1.8
	10	•					•			•	**1.3**
	11	•						•			2.0
	12	•						•		•	**1.3**
	13	•							•		1.8
	14	•							•	•	1.7
ALNS	15	•	•	•	•	•	•	•	•	•	**1.3**

Table 3: Simple LNS heuristics compared to the full adaptive LNS with dynamic weight adjustment. The first column shows if the configuration must be considered as an LNS or an ALNS heuristic. The second column is the configuration number, columns three to five indicate which removal heuristics were used. Columns six to ten indicate which insertion heuristics were used. Column eleven states if noise was added to the objective function during insertion of requests (in this case noise was added to the objective function in 50% of the insertions for the simple configurations 1-14 while in configuration 15 the number of noise-insertions was controlled by the adaptive method). Column twelve shows the average performance of the different heuristics. As an example, in configuration four we used random removal together with the regret-2 insertion heuristic and we applied noise to the objective value. This resulted in a set of solutions whose objective values on average were 3.2% above the best solutions found during the whole experiment.

basic insertion heuristic nearly performs as well as the regret heuristics when used in a LNS framework. This is surprising seen in the light of Table 1 where the basic insertion heuristic performed particularly badly. This observation may indicate that the LNS method is relatively robust with respect to the insertion method used.

The last row of the table shows the performance of ALNS. As one can see, it is on par with the two best simple approaches, but not better, which at first may seem disappointing. The results show though, that the adaptive mechanism is able to find a sensible set of weights, and it is our hypothesis that the ALNS heuristic is more robust than the simpler LNS heuristics. That is, the simple configuration may fail to produce good solutions on other types of problems, while the ALNS heuristic continues to perform well. One of the purposes of the experiments in Section 4.4 is to confirm or disprove this hypothesis.

4.4 Results

This section provides computational experiments conducted to test the performance of the heuristic. There are three major objectives for this section:

1. To compare the ALNS heuristic to a simple LNS heuristic that only contains one removal and one insertion heuristic.
2. To determine if certain problem properties influence the (A)LNS heuristics ability to find good solutions.
3. To compare the ALNS heuristic with state-of-the-art PDPTW heuristics from the literature.

In order to clarify if the ALNS heuristic is worthwhile compared to a simpler LNS heuristic we are going to show results for both the ALNS heuristic and the best simple LNS heuristic from Table 3. Configuration 12 was chosen as representative for the simple LNS heuristics as it performed slightly better than configuration 10. In the following sections we refer to the full and simple LNS heuristic as ALNS and LNS respectively.

All experiments were performed on a 1.5 GHz Pentium IV PC with 256 MB internal memory, running Linux. The implemented algorithm measures travel times and distances using double precision floating point numbers. The parameter setting found in Section 4.3.2 was used in all experiments unless otherwise stated.

4.4.1 Data sets

As the model considered in this paper is quite complicated, it is hard to find any benchmark instances that consider exactly the same model and objective function. The benchmark instances that come closest to the model considered in this paper are the instances constructed by Nanry and Barnes [15] and the instances constructed by Li and Lim [11]. Both data sets are single depot pickup and delivery problems with time windows, constructed from VRPTW problems. We are only reporting results on the data set proposed by Li and Lim, as the Nanry and Barnes instances are easy to solve due to their size.

The problem considered by Li and Lim were simpler than the one considered in this paper as: 1) it did not contain multiple depots; 2) all requests must be served; 3) all vehicles were assumed to be able to serve all requests. When solving the Li and Lim instances using the ALNS heuristic we set α to one and β to zero in our objective function. In section 4.5 we minimize the number of vehicles as first priority while we in section 4.4.2 only minimize the distance driven.

In order to test all aspects of the model proposed in this paper, we also introduce some new, randomly generated instances. These instances are described in section 4.4.3.

4.4.2 Comparing ALNS and LNS using the Li & Lim instances

This section compares the ALNS and LNS heuristics using the benchmark instances proposed by Li and Lim [11]. The data set contains 354 instances with between 100 and 1000 locations. The data set can be downloaded from [25].

In this section we use the distance driven as our objective even though vehicle minimization is the standard primary objective for these instances. The reason for this decision is that distance minimization makes comparison of the heuristics easier and distance minimization is the original objective of the proposed heuristic. The number of vehicles available for serving the requests is set to the minimum values reported by Li and Lim in [11] and on their web page which unfortunately no longer is on-line.

The heuristics were applied 10 times to each instance with 400 or less locations and 5 times to each instance with more than 400 locations. The experiments are summarized in Table 4.

		Best known solutions		Avg. gap (%)		Average time (s)		Fails	
#locations	#problems	ALNS	LNS	ALNS	LNS	ALNS	LNS	ALNS	LNS
100	56	52	50	0.19	0.50	49	55	0	0
200	60	49	15	0.72	1.41	305	314	0	0
400	60	52	6	2.36	4.29	585	752	0	0
600	60	54	5	0.93	3.20	1069	1470	0	0
800	60	46	5	1.73	3.27	2025	3051	0	2
1000	58	47	4	2.26	4.22	2916	5252	0	1

Table 4: Summary of results obtained on Li and Lim instances [11]. The first column gives the problem size; the next column indicates the number of problems in the data set of the particular size. The rest of the table consists of four major columns, each divided into two sub columns, one for the ALNS and one for LNS. The column *Best known solutions* indicates for how many problems the best known solution was identified. The best known solution is either the solution reported by Li and Lim or the best solution identified by the (A)LNS heuristics depending on which is best. The next column indicates how far the average solution is from best known solution. This number is averaged over all problems of a particular size. The next column shows how long the heuristic on average spends to solve a problem. The last column shows the number of times the heuristic failed to find a solution where all request are served by the given number of vehicles in all the attempts to solve a particular problem.

The results show that the ALNS heuristic on all four terms performs better than the LNS heuristic. One also notices that the ALNS heuristic becomes even more attractive as the problem size increases. It may seem odd that the LNS heuristic spends more time compared to the ALNS heuristic when they both perform the same number of LNS iterations. The reason for this behavior is that the Shaw removal heuristic used by the LNS heuristic is more time consuming compared to the two other removal heuristics.

4.4.3 New instances

This section provides results on randomly generated PDPTW instances that contain features of the model that were not used in the Li and Lim benchmark problems considered in Section 4.4.2. These features are: multiple depots, routes with different start and end terminals and *special* requests that only can be served by a certain subset of the vehicles. When solving these instances we set $\alpha = \beta = 1$ in the objective function so that distance and time are weighted equally in the objective function. We do not perform vehicle minimization as the vehicles are inhomogeneous.

Three types of geographical distributions of requests are considered: problems with locations distributed uniformly in the plane, problems with locations distributed in 10 clusters and problems with 50% of the locations are put in 10 clusters and 50% of the locations distributed uniformly. These three types of problems were inspired by Solomon's VRPTW benchmark problems [26], and the problems are similar to the R, the C and the RC Solomon problems respectively. We consider problems with 50, 100, 250 and 500 requests, all problems are multi depot problems. For each problem size we generated 12 problems as we tried every combination of the three problem features shown below:

- Route type: 1) A route starts and ends at the same location, 2) a route starts and ends at different locations.
- Request type: 1) All requests are normal requests, 2) 50% of the requests are *special* requests. The special requests can only be served by a subset of the vehicles. In the test problems each special request could only be served by between 30% to 60% of the vehicles.
- Geographical distributions: 1) Uniform, 2) Clustered, 3) Semi-clustered.

The instances can be downloaded from www.diku.dk/~sropke. The heuristics were tested by applying them to each of the 48 problems 10 times. Table 5 shows a summary of the results found. In the table we list for how many problems the two heuristics find the best known solution. The best known solution is simply the best solution found throughout this experiment.

We observe the same tendencies as in Table 4; ALNS is still superior to LNS, but one notices that the gap in solution quality between the two methods are smaller for this set of instances while the difference in running time is larger compared to the results on the Li and Lim instances. One also notices that it seems harder to solve small instances of this problem class compared to the Li and Lim instances.

		Best known solutions		Avg. gap (%)		Average time (s)	
#requests	#problems	ALNS	LNS	ALNS	LNS	ALNS	LNS
50	12	8	5	1.44	1.86	23	34
100	12	11	1	1.54	2.18	83	142
250	12	7	5	1.39	1.62	577	1274
500	12	9	3	1.18	1.32	3805	8146
Sum:	48	35	14	5.55	6.98	4488	9596

Table 5: Summary of results obtained on new instances. The captions of the table should be interpreted as in Table 4. The last row sums each column. Notice that the size of the problems in this table is given as number of requests and not the number of locations.

Table 6 summarizes how the problem features influence the average solution quality. These results show that the clustered problems are the hardest to solve, while the uniformly distributed instances are the easiest. The results also indicate that special requests make the problem slightly harder to solve. The route type experiments compare the situation where routes start and end at the same location (the typical situation considered in the literature) to the situation where each route starts and ends at different locations. Here we expect the last case to be the easiest to solve, as we by having different start and end positions for our routes, gain information about the area the route most likely should cover. The results in Table 6 confirm these expectations.

In addition to investigate the question of how the model features influence the average solution quality obtained by the heuristics we also want to know if the presence of some features could make LNS behave better than ALNS. For the considered features the answer is negative.

Feature	ALNS	LNS
Distribution: Uniform	1.04%	1.50%
Distribution: Clustered	1.89%	2.09%
Distribution: Semi-clustered	1.23%	1.64%
Normal Requests	1.24%	1.47%
Special Requests	1.54%	2.02%
Start of route = end of route	1.59%	2.04%
Start of route $\neq$ end of route	1.19%	1.45%

Table 6: Summary of the influence of certain problem features on the heuristic solutions. The two columns correspond to the two heuristic configurations. Each row shows the average solution quality for each feature. The average solution quality is defined as the average of the average gap for all instances with a specific feature. To be more precise, the solution quality is calculated using the formula: $q(h) = \frac{1}{|F|}\sum_{i\in F}\left(\frac{1}{10}\sum_{j=1}^{10}\frac{c(i,j,h)-c'(i)}{c'(i)}\right)$ where F is the set of instances with a specific feature, $c'(i)$ is the cost of the best known solution to instance i and $c(i,j,h)$ is the cost obtained in the jth experiment on instance i using heuristic h.

4.5 Comparison to existing heuristics

This section compares the ALNS heuristics to existing heuristics for the PDPTW. The comparison is performed using the benchmark instances proposed by Li and Lim [11] that also were used in Section 4.4.2. When PDPTW problems have been solved in the literature, the primary objective has been to minimize the number of vehicles used while the secondary objective has been to minimize the traveled distance. For this purpose we use the vehicle minimization algorithm described in Section 3.7. The ALNS heuristic was applied 10 times to each instance with 200 or less locations and 5 times to each instance with more than 200 locations. The experiments are summarized in Tables 7, 8 and 9. It should be noted that it was necessary to decrease the w parameter and increase the c parameter when the instances with 1000 locations were solved in order to get reasonable solution quality. Apart from that, the same parameter setting has been used for all instances.

In the literature, four heuristics have been applied to the benchmark problems: the heuristic by Li and Lim [11], the heuristic by Bent and Van Hentenryck [2] and two commercial heuristics; a heuristic developed by SINTEF and a heuristic developed by TetraSoft A/S. Detailed results for the two last heuristics are not available but some results obtained using these heuristics can be found on a web page maintained by SINTEF [25]. The heuristic that has obtained the best overall solution quality so far is probably the one by Bent and Van Hentenryck [2] (shortened BH heuristic in the following), therefore the ALNS heuristic is compared to this heuristic in Table 7. The complete results from the BH heuristic can be found in [3]. The results given for the BH heuristic are the best obtained among 10 experiments (though for the 100 location instances only 5 experiments were performed). The *Avg. TTB* column shows the average time needed for the BH heuristic to obtain its best solution. For the ALNS heuristic we only list the time used in total as this heuristic - because of its simulated annealing component, the heuristic usually finds its best solution towards the end of the search. The BH heuristic was tested on a 1.2 GHz Athlon processor and the running times of the two heuristics should therefore be comparable (we believe that the Athlon processor is at most 20% slower than our computer). The results show that the ALNS heuristic overall dominates the BH heuristic, especially as the problem sizes increase. It is also clear that the ALNS heuristic is able to improve considerably on the previously best known solutions and that the vehicle minimization algorithm works very well despite its simplicity. The last two columns in Table 7 summarize the best results obtained using several experiments with different parameter settings, which show that the results obtained by ALNS actually can be improved even further.

Table 8 compares the results obtained by ALNS with the best known solutions from the literature. It can be seen that ALNS improves more than half of the solutions and achieves a solution that is at least as good as the previously best known solution for 80% of the problems.

The two afore mentioned tables only dealt with the best solutions found by the ALNS heuristic. Table 9 shows the average solution quality obtained by the heuristic. These numbers can be compared to those in Table 7. It is worth noticing that the average solution sometimes have a lower distance than the “best of 10 or 5” solution in table 7, this is the case in the last row. This is possible because the heuristic finds solutions that use more than the minimum number of vehicles and this usually makes solutions with shorter distances

#locations	Best known 2003 #veh.	Dist	BH best #veh.	Dist	Avg. TTB	Avg. time	ALNS best of 10 or 5 #veh.	Dist	Avg. time	ALNS best #veh.	Dist
100	402	58060	402	58062	68	3900	402	58060	66	402	56060
200	615	178380	614	180358	772	3900	606	180931	264	606	180419
400	1183	421215	1188	423636	2581	6000	1158	422201	881	1157	420396
600	1699	873850	1718	879940	3376	6000	1679	863442	2221	1664	860898
800	2213	1492200	2245	1480767	5878	8100	2208	1432078	3918	2181	1423063
1000	2698	2195755	2759	2225190	6174	8100	2652	2137034	5370	2646	2122922

Table 7: This table compares the ALNS heuristic to existing heuristics using the Li and Lim benchmark instances. Each row in the table corresponds to a set of problems with the same number of locations. Each of these problem sets contain between 56 and 60 instances (see Table 8). The first column indicates the number of locations in each problem; the next two columns give the total number of vehicles used and the total distance traveled in the previously best known solutions as listed on the SINTEF web page [25] in the summer of 2003. The next four columns show information about the solutions obtained by Bent and Van Hentenryck's heuristic [2]. The two columns *Avg. TTB* and *Avg. time* show the average time needed to reach the best solution and the average time spent on each instance, respectively. Both columns report the time needed to perform one experiment on one instance. The next three columns report the solutions obtained in the experiment with the ALNS heuristic where the heuristic was applied either 5 or 10 times to each problem. The last two columns report the best solutions obtained in several experiments with our ALNS heuristic and with various parameter settings. Note that Bent and Van Hentenryck in some cases have found slightly better results than reported on the SINTEF web page in 2003. This is the reason why the number of vehicles used by the BH heuristic for the 200 locations problems is smaller than in the best known solutions.

possible.

Overall, one can conclude that the ALNS heuristic must be considered as a state of the art heuristic for the PDPTW. The cost of the best solutions identified during the experiments are listed in Tables 10 to 15.

4.6 Computational tests conclusion

In Section 4.4 we stated three objectives for our computational experiments. The tests fulfilled these objectives as we saw that: 1) the adaptive LNS heuristic that combines several removal and construction heuristics displays superior performance compared to the simple LNS heuristic that only uses one insertion heuristic and one removal heuristic; 2) certain problem characteristics influence the performance of the LNS heuristic but we did not find that any characteristics could make the LNS heuristic perform better than the ALNS heuristic; 3) the LNS heuristic indeed is able to find good quality solutions in a reasonable amount of time, and the heuristic outperforms previously proposed heuristics.

The experiments also illustrate the importance of testing heuristics on large sets of problem instances as the

#locations	#problems	ALNS best of 10 or 5 $<$PB	$\leq$PB	ALNS best $<$PB	$\leq$PB
100	56	0	54	0	55
200	60	22	42	27	57
400	60	40	47	41	55
600	60	41	45	51	57
800	60	37	42	48	53
1000	58	50	54	51	55

Table 8: Comparison of the ALNS heuristic to the previously best known solutions. The table is grouped by problem size. The first column shows the problem size, the next column shows the number of problems of that size. The next two columns give additional information about the experiment where the ALNS heuristic was applied 5 or 10 times to each instance. The columns $<PB$ report how many times the best solution found by the ALNS heuristic was strictly better than the previously best known solution. The columns $\leq PB$ show how many times the best solution found by ALNS was at least as good as the previously best known solution. The last two columns show information about the best solutions obtained during experimentation with different parameter settings.

#locations	Avg. #veh.	Avg. Dist
100	403	58249
200	608	181707
400	1168	425817
600	1686	867930
800	2223	1432321
1000	2677	2129032

Table 9: The ALNS heuristic was applied 10 times to each problem with 200 or less locations and 5 times to each problem with more than 200 locations. The best solutions reported in Table 7 and 8 were of course not obtained in all experiments. This table shows the average number of vehicles and average distance traveled obtained. These numbers can be compared to the figures in Table 7

	R1		R2		C1		C2		RC1		RC2	
1	**19**	**1650.80**	**4**	**1253.23**	**10**	**828.94**	**3**	**591.56**	**14**	**1708.80**	**4**	**1406.94**
2	**17**	**1487.57**	**3**	**1197.67**	**10**	**828.94**	**3**	**591.56**	**12**	**1558.07**	**3**	**1374.27**
3	**13**	**1292.68**	**3**	**949.40**	**9**	**1035.35**	3	591.17	**11**	**1258.74**	**3**	**1089.07**
4	**9**	**1013.39**	**2**	**849.05**	**9**	**860.01**	**3**	**590.60**	**10**	**1128.40**	**3**	**818.66**
5	**14**	**1377.11**	**3**	**1054.02**	**10**	**828.94**	**3**	**588.88**	**13**	**1637.62**	**4**	**1302.20**
6	**12**	**1252.62**	**3**	**931.63**	**10**	**828.94**	**3**	**588.49**	**11**	**1424.73**	**3**	**1159.03**
7	**10**	**1111.31**	**2**	**903.06**	**10**	**828.94**	**3**	**588.29**	**11**	**1230.14**	**3**	**1062.05**
8	**9**	**968.97**	**2**	**734.85**	**10**	**826.44**	**3**	**588.32**	**10**	**1147.43**	**3**	**852.76**
9	**11**	**1208.96**	**3**	**930.59**	**9**	**1000.60**						
10	**10**	**1159.35**	**3**	**964.22**								
11	**10**	**1108.90**	**2**	**911.52**								
12	**9**	**1003.77**										

Table 10: Best results, 100 locations. The Li and Lim benchmark instances are divided into six sets: R1, R2, C1, C2, RC1 and RC2. Each of the major columns corresponds to one of these sets, the column at the left give the problem number. For each problem instance we report the number of vehicles and the distance traveled in the best solution obtained during experimentation. Bold numbers indicate best known solutions.

	R1		R2		C1		C2		RC1		RC2	
1	**20**	**4819.12**	**5**	**4073.10**	**20**	**2704.57**	**6**	**1931.44**	**19**	**3606.06**	**6**	**3605.40**
2	**17**	**4621.21**	**4**	**3796.00**	**19**	**2764.56**	**6**	**1881.40**	**15**	**3674.80**	**5**	**3327.18**
3	**15**	**3612.64**	**4**	**3098.36**	**17**	**3128.61**	**6**	**1844.33**	13	3178.17	**4**	**2938.28**
4	**10**	**3037.38**	**3**	**2486.14**	**17**	**2693.41**	**6**	**1767.12**	**10**	**2631.82**	**3**	**2887.97**
5	**16**	**4760.18**	**4**	**3438.39**	**20**	**2702.05**	**6**	**1891.21**	**16**	**3715.81**	**5**	**2776.93**
6	14	4178.24	**4**	**3201.54**	**20**	**2701.04**	**6**	**1857.78**	**17**	**3368.66**	**5**	**2707.96**
7	**12**	**3550.61**	**3**	**3135.05**	**20**	**2701.04**	**6**	**1850.13**	**14**	**3668.39**	4	3056.09
8	**9**	**2784.53**	**2**	**2555.40**	**20**	**2689.83**	**6**	**1824.34**	**13**	**3174.55**	**4**	**2399.95**
9	**14**	**4354.66**	**3**	**3930.49**	**18**	**2724.24**	**6**	**1854.21**	**13**	**3226.72**	**4**	**2208.49**
10	**11**	**3714.16**	**3**	**3344.08**	**17**	**2943.49**	**6**	**1817.45**	**12**	**2951.29**	**3**	**2550.56**

Table 11: Best results, 200 locations.

	R1		R2		C1		C2		RC1		RC2	
1	**40**	**10639.75**	8	9758.46	**40**	**7152.06**	**12**	**4116.33**	**36**	**9127.15**	**12**	**7471.01**
2	**31**	**10015.85**	**7**	**9496.64**	**38**	**8012.43**	**12**	**4144.29**	**31**	**8346.06**	**11**	**6303.36**
3	**23**	**8840.46**	**6**	**8116.53**	**33**	**8308.94**	**12**	**4431.75**	25	7387.40	**9**	**5438.20**
4	**16**	**6744.33**	**4**	**6649.78**	30	6878.00	12	4038.00	**19**	**5838.58**	**5**	**5322.43**
5	**29**	**10599.54**	**7**	**8574.84**	**40**	**7150.00**	**12**	**4030.63**	**33**	**8773.75**	**11**	**6120.13**
6	25	9525.45	**6**	**7995.06**	**40**	**7154.02**	**12**	**3900.29**	**31**	**8177.90**	**9**	**6479.56**
7	**19**	**8200.37**	**5**	**6928.61**	**40**	**7149.43**	**12**	**3962.51**	**29**	**7992.08**	**8**	**6361.26**
8	**14**	**5946.44**	**4**	**5447.40**	**39**	**7111.16**	**12**	**3844.45**	**27**	**7613.43**	**7**	**5928.93**
9	**24**	**9886.14**	**6**	**8043.20**	**36**	**7452.21**	**12**	**4188.93**	**26**	**8013.48**	**7**	**5303.53**
10	**21**	**8016.62**	**5**	**7904.77**	**35**	**7387.13**	**12**	**3828.44**	**24**	**7065.73**	**6**	**5760.78**

Table 12: Best results, 400 locations.

	R1		R2		C1		C2		RC1		RC2	
1	59	22838.65	**11**	**21945.30**	60	14095.64	**19**	**7977.98**	**53**	**17924.88**	**16**	**14817.72**
2	**45**	**20246.18**	**10**	**19666.59**	58	14379.53	**18**	**10277.23**	**44**	**16302.54**	**14**	**12758.77**
3	**37**	**18073.14**	**8**	**15609.96**	**50**	**14683.43**	**17**	**8728.30**	**36**	**14060.31**	**10**	**12812.67**
4	**28**	**13269.71**	**6**	**10819.45**	**47**	**13648.03**	**17**	**8041.97**	**25**	**10950.52**	**7**	**10574.87**
5	**38**	**22562.81**	**9**	**19567.41**	**60**	**14086.30**	**19**	**8047.37**	**47**	**16742.55**	**14**	**13009.52**
6	**32**	**20641.02**	**8**	**17262.96**	**60**	**14090.79**	**19**	**8094.11**	**44**	**16894.37**	**13**	**12643.98**
7	**25**	**17162.90**	**6**	**15812.42**	**60**	**14083.76**	**19**	**7998.18**	**39**	**15394.87**	**11**	**12007.65**
8	**19**	**11957.59**	**5**	**10950.90**	**59**	**14554.27**	**18**	**7579.93**	**36**	**15154.79**	**10**	**12163.43**
9	**32**	**21423.05**	**8**	**18799.36**	**54**	**14706.12**	**18**	**9501.00**	**35**	**15134.24**	**9**	**13768.01**
10	**27**	**18723.13**	**7**	**17034.63**	**53**	**14879.30**	**17**	**8019.94**	**31**	**13925.51**	**8**	**12016.94**

Table 13: Best results, 600 locations.

	R1		R2		C1		C2		RC1		RC2	
1	**80**	**39315.92**	**15**	**33816.90**	**80**	**25184.38**	**24**	**11687.06**	**67**	**32268.95**	**20**	**23289.40**
2	**59**	**34370.37**	**12**	**32575.97**	78	26062.17	**24**	**14358.92**	57	28395.39	**18**	**21786.62**
3	**44**	**29718.09**	**10**	**25310.53**	**65**	**25918.45**	**24**	**13198.29**	50	24354.36	**16**	**16586.31**
4	**25**	**21197.65**	**7**	**19506.42**	**60**	**22970.88**	**23**	**13376.82**	**35**	**18241.91**	**12**	**14122.05**
5	**50**	**39046.06**	**12**	**32634.29**	**80**	**25211.22**	25	12329.80	61	30995.48	**18**	**20292.92**
6	**42**	**33659.50**	**10**	**27870.80**	**80**	**25164.25**	**24**	**12702.87**	58	28568.61	**16**	**21088.57**
7	**32**	**27294.19**	**8**	**25077.85**	**80**	**25158.38**	**25**	**11855.86**	54	28164.41	**15**	**19695.96**
8	**21**	**19570.21**	**5**	**19256.79**	**78**	**25348.45**	**24**	**11482.88**	**49**	**26150.65**	**13**	**19009.33**
9	**42**	**36126.69**	**10**	**30791.77**	**73**	**25541.94**	**24**	**11629.61**	**47**	**24930.70**	**12**	**19003.68**
10	**32**	**30200.86**	**9**	**28265.24**	**71**	**25712.12**	**24**	**11578.58**	**42**	**24271.52**	**10**	**19766.78**

Table 14: Best results, 800 locations.

	R1		R2		C1		C2		RC1		RC2	
1	**100**	**56903.88**	**19**	**45422.58**	**100**	**42488.66**	**30**	**16879.24**	85	48702.83	**22**	**35073.70**
2	**80**	**49652.10**	**15**	**47824.44**	**95**	**43870.19**	**31**	**18980.98**	**73**	**45135.70**	**21**	**30932.74**
3	**54**	**42124.44**	**11**	**39894.32**	**82**	**42631.11**	**30**	**17772.49**	**55**	**35475.72**	**16**	**28403.51**
4	**28**	**32133.36**	**8**	**28314.95**	**74**	**39443.00**	**29**	**18089.93**	**40**	**27747.04**	**12**	**23083.20**
5	**61**	**59135.86**	**14**	**53209.98**	100	42477.41	**31**	**17137.53**	**76**	**49816.18**	**18**	**34713.96**
6	**50**	**48637.63**	**12**	**43792.11**	**101**	**42838.39**	**31**	**17198.01**	**69**	**44469.08**	**17**	**31485.26**
7	**37**	**38936.54**	**9**	**36728.20**	**100**	**42854.99**	**31**	**19117.67**	**64**	**41413.16**	**17**	**29639.63**
8	**26**	**29452.32**	**7**	**26278.09**	**98**	**42951.56**	**30**	**17018.63**	**60**	**40590.17**	-	-
9	**50**	**52223.15**	**13**	**48447.49**	**92**	**42391.98**	**31**	**17565.95**	**57**	**39587.85**	-	-
10	**40**	**46218.35**	**11**	**44155.66**	**90**	**42435.16**	**29**	**17425.55**	52	36195.00	**12**	**29402.90**

Table 15: Best results, 1000 locations. Two entries are missing as the corresponding problem instances no longer exist.

difference between LNS and ALNS only really becomes apparent when we consider large instances. Note that the problems that need to be solved in the real world often have dimensions comparable to or greater than the biggest problems solved in this paper.

Finally the computational experiments performed in Section 4.3.3 indicated that a simple LNS heuristic seems to be more sensitive to the choices of removal heuristic compared to the choices of insertion heuristics. It would be interesting to see if this holds in general for other problems as well.

5 Conclusion

This paper presented an extension to the large neighborhood search and the ruin and recreate heuristic called adaptive LNS. The heuristic was tested on the pickup and delivery problem with time windows achieving good results in a reasonable amount of time. The idea of combining several sub heuristics in the same search proved to be successful.

As the proposed model is quite general would be interesting to examine if the model and heuristic can be used to solve other vehicle routing problems. We are currently working on this topic and the results are very promising as the heuristic has been able to discover new best solutions to standard benchmarks for vehicle routing problems with time windows and multi-depot vehicle routing problems and other vehicle routing problems [16], [20].

It would also be interesting to apply the ideas presented in this paper to other combinatorial optimization problems. The adaptive LNS framework is easily applicable to most problems, taking advantage of the numerous robust and fast construction heuristics designed during the last decades for various optimization problems.

6 Acknowledgments

The authors wish to thank Jakob Birkedal Nielsen for helpful discussions during the development of the heuristic presented in this paper and Emilie Danna for valuable criticism of the paper. Furthermore we would like to thank the three anonymous referees for valuable comments and corrections.

7 Appendix

Tables 16 to 21 show detailed information about the solutions found during the experiment described in Section 4.5.

References

[1] R.K. Ahuja, O. Ergun, J.B. Orlin, A.P. Punnen, *A survey of Very Large Scale Neighborhood Search Techniques*, Discrete Applied Mathematics **123** (2002), 75-102.

[2] R. Bent, P. Van Hentenryck, *A Two-Stage Hybrid Algorithm for Pickup and Delivery Vehicle Routing Problems with Time Windows*, Proceedings of the International Conference on Constraint Programming (CP-2003), Lecture Notes in Computer Science **2833**, 123-137.

[3] R. Bent, P. Van Hentenryck, *A Two-Stage Hybrid Algorithm for Pickup and Delivery Vehicle Routing Problems with Time Windows*, Appendix, available at http://www.cs.brown.edu/people/rbent/pickup-appendix.ps.

[4] R. Bent, P. Van Hentenryck, *A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows*, To appear in Transportation Science.

[5] J.-F. Cordeau, G. Laporte, A. Mercier, *A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows*, Journal of the Operational Research Society **52** (2001), 928-936.

[6] T.H. Cormen, C.E., Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms, Second Edition*, MIT Press, 2001.

[7] G. Desaulniers, J. Desrosiers, A. Erdmann, M.M. Solomon, F. Soumis, *The VRP with Pickup and Delivery*, in P. Toth and D. Vigo (eds.): The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications **9** (2002), SIAM, Philadelphia, 225-242.

[8] Y. Dumas, J. Desrosiers, F. Soumis, *The pickup and delivery problem with time windows*, European Journal of Operational Research **54** (1991), 7-22.

[9] M. Gendreau, F. Guertin, J.-Y. Potvin, R. Séguin, *Neighborhood Search Heuristics for a Dynamic Vehicle Dispatching Problem with Pick-ups and Deliveries*, Tech. Rep. CRT-98-10, Centre de Recherche sur les Transport, Universite de Montreal.

[10] H. C. Lau, Z. Liang, *Pickup and Delivery with Time Windows : Algorithms and Test Case Generation*, The 13th IEEE Conference on Tools with Artificial Intelligence, ICTAI-2001, Dallas, USA, 333-340.

[11] H. Li, A. Lim, *A Metaheuristic for the Pickup and Delivery Problem with Time Windows*, The 13th IEEE Conference on Tools with Artificial Intelligence, ICTAI-2001, Dallas, USA, 160-170.

[12] H. Lim, A. Lim, B. Rodrigues, *Solving the Pick up and Delivery Problem with Time Windows using "Squeaky Wheel" Optimization with Local Search*, Technical Report 2002, Singapore Management University, Paper No. 7-2002

[13] S. Martello, P. Toth, *An algorithm for the generalized assignment problem*, Operational Research '81, J.P. Brans (editor), North-Holland, New York (1981).

[14] N. Mladenović, P. Hansen, *Variable neighborhood search*, Computers and Operations Research 24, No. 11 (1997), 1097-1100.

[15] W.P. Nanry, J.W. Barnes, *Solving the pickup and delivery problem with time windows using reactive tabu search*, Transportation Research Part B **34** (2000), 107-121.

[16] D. Pisinger, S. Ropke, *A general heuristic for vehicle routing problems*, accepted for publication, Computers and Operations Research (2005).

[17] J.-Y. Potvin, J.-M. Rousseau, *A parallel route building algorithm for the vehicle routing and scheduling problem with time windows*, European Journal of Operational Research **66** (1993), 331-340.

[18] S. Ropke, D. Pisinger, *An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows*, Technical Report, Department of Computer Science, University of Copenhagen 2004.

[19] S. Ropke, *Heuristics for the Multi-Vehicle Pickup and Delivery Problem with Time Windows*, Masters Thesis 01-11-8, Department of Computer Science, Univerity of Copenhagen, 2002.

[20] S. Ropke, D. Pisinger, *A Unified Heuristic for a Large Class of Vehicle Routing Problems with Backhauls*, to appear in European Journal of Operational Research.

[21] P. Shaw, *A new local search algorithm providing high quality solutions to vehicle routing problems*, Technical report, Department of Computer Science, University of Strathclyde, Scotland, 1997.

[22] P. Shaw, *Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems*, Proceedings CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming), 1998.

[23] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, G. Dueck, *Record Breaking Optimization Results Using the Ruin and Recreate Principle*, Journal of Computational Physics **159** (2000), 139-171.

[24] M. Sigurd , D. Pisinger, M. Sig, *The Pickup and Delivery Problem With Time Windows and Precedences*, Transportation Science **38** (2004), 197-209.

[25] SINTEF, *Vehicle Routing and Travelling Salesperson Problems*, Web page: http://www.sintef.no/static/am/opti/projects/top/vrp/benchmarks.html

[26] M.M. Solomon, *Algorithms for the vehicle routing and scheduling problems with time window constraints*, Operations Research **35** (1987), 254-265.

[27] M.A Trick, *A Linear Relaxation Heuristic for the Generalized Assignment Problem*, Naval Research Logistics **39** (1992), 137-152.

[28] P. Toth, D. Vigo, *An Overview of Vehicle Routing Problems*, In P. Toth and D. Vigo (eds.): The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications **9** (2002), SIAM, Philadelphia, 1-26.

[29] H. Xu, Z.-L. Chen, S. Rajagopal, S. Arunapuram, *Solving a Practical Pickup and Delivery Problem*, Transportation Science **37** (2003), 347-364.

	Best known			FULL					LNS best known	
	veh.	cost	References	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. time (s)	veh.	cost
LR101	**19**	**1650.8**	LL	1650.80	19.0	**1650.80**	**19**	40	**19**	**1650.80**
LR102	**17**	**1487.57**	LL	1487.57	17.0	**1487.57**	**17**	47	**17**	**1487.57**
LR103	**13**	**1292.68**	LL	1292.68	13.0	**1292.68**	**13**	45	**13**	**1292.68**
LR104	**9**	**1013.39**	LL	1013.39	9.0	**1013.39**	**9**	26	**9**	**1013.39**
LR105	**14**	**1377.11**	LL	1377.11	14.0	**1377.11**	**14**	40	**14**	**1377.11**
LR106	**12**	**1252.62**	LL	1252.62	12.0	**1252.62**	**12**	41	**12**	**1252.62**
LR107	**10**	**1111.31**	LL	1111.31	10.0	**1111.31**	**10**	44	**10**	**1111.31**
LR108	**9**	**968.97**	LL	968.97	9.0	**968.97**	**9**	25	**9**	**968.97**
LR109	**11**	**1208.96**	SAM	1208.96	11.0	**1208.96**	**11**	41	**11**	**1208.96**
LR110	**10**	**1159.35**	LL	1159.35	10.0	**1159.35**	**10**	35	**10**	**1159.35**
LR111	**10**	**1108.9**	LL	1108.90	10.0	**1108.90**	**10**	44	**10**	**1108.90**
LR112	**9**	**1003.77**	LL	1003.77	9.0	**1003.77**	**9**	27	**9**	**1003.77**
LC101	**10**	**828.94**	LL	828.94	10.0	**828.94**	**10**	43	**10**	**828.94**
LC102	**10**	**828.94**	LL	828.94	10.0	**828.94**	**10**	44	**10**	**828.94**
LC103	**9**	**1035.35**	BH	1037.77	9.0	**1035.35**	**9**	49	**9**	**1035.35**
LC104	**9**	**860.01**	SAM	860.15	9.0	**860.01**	**9**	63	**9**	**860.01**
LC105	**10**	**828.94**	LL	828.94	10.0	**828.94**	**10**	41	**10**	**828.94**
LC106	**10**	**828.94**	LL	828.94	10.0	**828.94**	**10**	42	**10**	**828.94**
LC107	**10**	**828.94**	LL	828.94	10.0	**828.94**	**10**	43	**10**	**828.94**
LC108	**10**	**826.44**	LL	826.44	10.0	**826.44**	**10**	46	**10**	**826.44**
LC109	**9**	**1000.6**	BH	1000.60	9.0	**1000.60**	**9**	35	**9**	**1000.60**
LRC101	**14**	**1708.8**	LL	1708.80	14.0	**1708.80**	**14**	38	**14**	**1708.80**
LRC102	**12**	**1558.07**	SAM	1558.07	12.0	**1558.07**	**12**	41	**12**	**1558.07**
LRC103	**11**	**1258.74**	LL	1258.74	11.0	**1258.74**	**11**	43	**11**	**1258.74**
LRC104	**10**	**1128.4**	LL	1128.40	10.0	**1128.40**	**10**	52	**10**	**1128.40**
LRC105	**13**	**1637.62**	LL	1637.62	13.0	**1637.62**	**13**	42	**13**	**1637.62**
LRC106	**11**	**1424.73**	SAM	1424.73	11.0	**1424.73**	**11**	42	**11**	**1424.73**
LRC107	11	1230.15	LL	1230.14	11.0	**1230.14**	**11**	43	**11**	**1230.14**
LRC108	**10**	**1147.43**	SAM	1147.43	10.0	**1147.43**	**10**	25	**10**	**1147.43**
LR201	**4**	**1253.23**	SAM	1253.23	4.0	**1253.23**	**4**	69	**4**	**1253.23**
LR202	**3**	**1197.67**	LL	1197.67	3.0	**1197.67**	**3**	60	**3**	**1197.67**
LR203	**3**	**949.4**	LL	949.40	3.0	**949.40**	**3**	98	**3**	**949.40**
LR204	**2**	**849.05**	LL	849.05	2.0	**849.05**	**2**	181	**2**	**849.05**
LR205	**3**	**1054.02**	LL	1054.02	3.0	**1054.02**	**3**	58	**3**	**1054.02**
LR206	**3**	**931.63**	LL	931.63	3.0	**931.63**	**3**	86	**3**	**931.63**
LR207	**2**	**903.06**	LL	903.06	2.0	**903.06**	**2**	187	**2**	**903.06**
LR208	**2**	**734.85**	LL	734.85	2.0	**734.85**	**2**	285	**2**	**734.85**
LR209	**3**	**930.59**	SAM	930.59	3.0	**930.59**	**3**	73	**3**	**930.59**
LR210	**3**	**964.22**	LL	964.22	3.0	**964.22**	**3**	77	**3**	**964.22**
LR211	**2**	**911.52**	SAM	906.69	2.2	**911.52**	**2**	126	**2**	**911.52**
LC201	**3**	**591.56**	LL	591.56	3.0	**591.56**	**3**	36	**3**	**591.56**
LC202	**3**	**591.56**	LL	591.56	3.0	**591.56**	**3**	59	**3**	**591.56**
LC203	**3**	**585.56**	LL	591.17	3.0	591.17	3	81	3	591.17
LC204	**3**	**590.6**	SAM	590.60	3.0	**590.60**	**3**	141	**3**	**590.60**
LC205	**3**	**588.88**	LL	588.88	3.0	**588.88**	**3**	48	**3**	**588.88**
LC206	**3**	**588.49**	LL	588.49	3.0	**588.49**	**3**	60	**3**	**588.49**
LC207	**3**	**588.29**	LL	588.29	3.0	**588.29**	**3**	62	**3**	**588.29**
LC208	**3**	**588.32**	LL	588.32	3.0	**588.32**	**3**	69	**3**	**588.32**
LRC201	**4**	**1406.94**	SAM	1406.94	4.0	**1406.94**	**4**	38	**4**	**1406.94**
LRC202	**3**	**1374.27**	LL	1387.74	3.8	1374.79	3	82	**3**	**1374.27**
LRC203	**3**	**1089.07**	SAM	1089.07	3.0	**1089.07**	**3**	69	**3**	**1089.07**
LRC204	**3**	**818.66**	SAM	818.66	3.0	**818.66**	**3**	173	**3**	**818.66**
LRC205	**4**	**1302.2**	LL	1302.20	4.0	**1302.20**	**4**	75	**4**	**1302.20**
LRC206	**3**	**1159.03**	SAM	1337.75	3.0	**1159.03**	**3**	48	**3**	**1159.03**
LRC207	**3**	**1062.05**	SAM	1062.05	3.0	**1062.05**	**3**	66	**3**	**1062.05**
LRC208	**3**	**852.76**	LL	852.76	3.0	**852.76**	**3**	88	**3**	**852.76**
Tot.	402	58054		58249.42	403.00	58060.03	402	3680	402	58059.50
Avg.								66		
< PB						1				1
<= PB						54				55
#B		55				54				55

Table 16: Results on 100-customer problems solved with vehicle minimization as primary objective. The first column contains the name of the problem, columns two to four show information about the previously best known solutions. Columns two and three give the number of vehicles in the solution and the total traveled distance. Column four refers to the method that first found the solution (LL: Li and Lim [11], BH: Bent and Van Hentenryck [2], SAM: SINTEF heuristic, TS: TetraSoft A/S heuristic). The next five columns show information about the solutions obtained by the ALNS LNS heuristic. The first two of these columns show the average distance traveled and the average number of vehicles (averaged over the 10 experiments performed). The two next column display the the best solution obtained in the 10 experiments. The column *avg. time* displays the average time needed to perform one experiment in seconds. The two last columns show the best results obtained during experimentation with various parameter settings. The last 5 columns provide some summary information. The *Tot.* and *Avg.* rows respectively sums and averages entries in the columns. The *<PB* row indicates how many solutions that are better than the previously best known solution and the *<=PB* row indicates how many solution that are at least as good as the previously best known solution. *#B* reports the number of overall best known solutions that were obtained. Best known solutions are marked with bold font.

	Best known			FULL					LNS best known	
	veh.	cost	References	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. time (s)	veh.	cost
LR1_2_1	**20**	**4819.12**	LL	4819.12	20.0	**4819.12**	**20**	137	**20**	**4819.12**
LR1_2_2	17	4666.09	BH	4625.99	17.0	**4621.21**	**17**	149	**17**	**4621.21**
LR1_2_3	**15**	**3612.64**	TS	3626.13	15.0	**3612.64**	**15**	173	**15**	**3612.64**
LR1_2_4	10	3146.06	BH	3088.07	10.0	3058.12	10	228	**10**	**3037.38**
LR1_2_5	**16**	**4760.18**	BH	4852.41	16.0	**4760.18**	**16**	136	**16**	**4760.18**
LR1_2_6	**14**	**4175.16**	BH	4261.23	14.0	4184.80	14	164	14	4178.24
LR1_2_7	12	3851.36	BH	3580.94	12.0	3551.47	12	173	**12**	**3550.61**
LR1_2_8	9	2871.67	BH	2823.91	9.0	**2784.53**	**9**	226	**9**	**2784.53**
LR1_2_9	14	4411.54	BH	4438.36	14.0	**4354.66**	**14**	144	**14**	**4354.66**
LR1_2_10	11	3744.95	BH	3787.23	11.0	3741.29	11	146	**11**	**3714.16**
LRC1_2_1	**19**	**3606.06**	SAM	3606.06	19.0	**3606.06**	**19**	136	**19**	**3606.06**
LRC1_2_2	15	3681.36	BH	3684.82	15.0	**3674.80**	**15**	143	**15**	**3674.80**
LRC1_2_3	**13**	**3161.75**	BH	3211.85	13.0	3178.17	13	183	13	3178.17
LRC1_2_4	10	2655.27	BH	2660.26	10.0	2641.67	10	284	**10**	**2631.82**
LRC1_2_5	**16**	**3715.81**	BH	3718.57	16.0	3716.72	16	141	**16**	**3715.81**
LRC1_2_6	**17**	**3368.66**	SAM	3372.68	17.0	3368.74	17	141	**17**	**3368.66**
LRC1_2_7	15	3417.16	BH	3525.21	14.7	**3668.39**	**14**	140	**14**	**3668.39**
LRC1_2_8	14	3087.62	BH	3220.69	13.2	**3174.55**	**13**	144	**13**	**3174.55**
LRC1_2_9	14	3129.65	BH	3259.40	13.1	**3226.72**	**13**	140	**13**	**3226.72**
LRC1_2_10	13	2833.85	BH	2968.69	12.1	2967.70	12	156	**12**	**2951.29**
LC1_2_1	**20**	**2704.57**	LL	2704.57	20.0	**2704.57**	**20**	146	**20**	**2704.57**
LC1_2_2	**19**	**2764.56**	LL	2764.56	19.0	**2764.56**	**19**	141	**19**	**2764.56**
LC1_2_3	17	3134.08	BH	3142.99	17.0	3136.42	17	155	**17**	**3128.61**
LC1_2_4	17	2698.73	TS	2711.42	17.0	2704.41	17	209	**17**	**2693.41**
LC1_2_5	**20**	**2702.05**	LL	2702.05	20.0	**2702.05**	**20**	137	**20**	**2702.05**
LC1_2_6	**20**	**2701.04**	LL	2701.04	20.0	**2701.04**	**20**	133	**20**	**2701.04**
LC1_2_7	**20**	**2701.04**	LL	2701.04	20.0	**2701.04**	**20**	139	**20**	**2701.04**
LC1_2_8	**20**	**2689.83**	LL	2689.83	20.0	**2689.83**	**20**	145	**20**	**2689.83**
LC1_2_9	**18**	**2724.24**	LL	2724.24	18.0	**2724.24**	**18**	157	**18**	**2724.24**
LC1_2_10	18	2741.56	LL	2967.24	17.0	**2943.49**	**17**	104	**17**	**2943.49**
LR2_2_1	**5**	**4073.1**	SAM	4110.08	5.0	**4073.10**	**5**	230	**5**	**4073.10**
LR2_2_2	**4**	**3796**	SAM	4194.32	4.0	4113.64	4	249	**4**	**3796.00**
LR2_2_3	**4**	**3098.36**	SAM	3209.80	4.0	**3098.36**	**4**	696	**4**	**3098.36**
LR2_2_4	3	2487.65	TS	2495.48	3.0	2491.87	3	1191	**3**	**2486.14**
LR2_2_5	**4**	**3438.39**	SAM	3440.71	4.0	3439.40	4	207	**4**	**3438.39**
LR2_2_6	**4**	**3201.54**	LL	3204.44	4.0	3201.86	4	499	**4**	**3201.54**
LR2_2_7	3	3190.75	LL	3216.40	3.0	**3135.05**	**3**	521	**3**	**3135.05**
LR2_2_8	3	2187.01	TS	2613.39	2.0	2559.70	2	1114	**2**	**2555.40**
LR2_2_9	4	3198.44	SAM	3272.31	3.9	**3930.49**	**3**	425	**3**	**3930.49**
LR2_2_10	3	3377.45	SAM	3387.47	3.0	3360.74	3	342	**3**	**3344.08**
LRC2_2_1	6	3690.1	BH	3722.20	6.0	3622.11	6	117	**6**	**3605.40**
LRC2_2_2	6	2666.01	BH	3403.75	5.0	**3327.18**	**5**	201	**5**	**3327.18**
LRC2_2_3	4	3141.28	SAM	3138.84	4.0	2965.88	4	323	**4**	**2938.28**
LRC2_2_4	4	2190.88	TS	3006.86	3.0	2891.10	3	993	**3**	**2887.97**
LRC2_2_5	**5**	**2776.93**	BH	2786.49	5.0	2782.83	5	302	**5**	**2776.93**
LRC2_2_6	**5**	**2707.96**	SAM	2713.57	5.0	2710.14	5	302	**5**	**2707.96**
LRC2_2_7	**4**	**3050.03**	BH	3140.57	4.0	3056.09	4	217	4	3056.09
LRC2_2_8	4	2401.84	BH	2409.16	4.0	2404.09	4	286	**4**	**2399.95**
LRC2_2_9	4	2209.54	SAM	2214.37	4.0	2210.88	4	410	**4**	**2208.49**
LRC2_2_10	3	2699.55	BH	2558.03	3.1	2551.67	3	467	**3**	**2550.56**
LC2_2_1	**6**	**1931.44**	SAM	1931.44	6.0	**1931.44**	**6**	100	**6**	**1931.44**
LC2_2_2	**6**	**1881.4**	LL	1881.40	6.0	**1881.40**	**6**	157	**6**	**1881.40**
LC2_2_3	**6**	**1844.33**	SAM	1845.57	6.0	1844.66	6	234	**6**	**1844.33**
LC2_2_4	**6**	**1767.12**	LL	1772.02	6.0	1768.22	6	427	**6**	**1767.12**
LC2_2_5	**6**	**1891.21**	LL	1891.21	6.0	**1891.21**	**6**	121	**6**	**1891.21**
LC2_2_6	**6**	**1857.78**	SAM	1857.93	6.0	**1857.78**	**6**	150	**6**	**1857.78**
LC2_2_7	**6**	**1850.13**	SAM	1850.60	6.0	**1850.13**	**6**	151	**6**	**1850.13**
LC2_2_8	**6**	**1824.34**	LL	1825.88	6.0	1824.73	6	193	**6**	**1824.34**
LC2_2_9	**6**	**1854.21**	SAM	1854.43	6.0	**1854.21**	**6**	193	**6**	**1854.21**
LC2_2_10	**6**	**1817.45**	LL	1818.04	6.0	**1817.45**	**6**	245	**6**	**1817.45**
Tot.	615	178380		181707.35	608.10	180930.62	606	15815	606	180418.58
Avg.								264		
< PB						22				27
<= PB						42				57
#B		33				31				57

Table 17: Results on 200-customer problems

	Best known			FULL					LNS best known	
	veh.	cost	References	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. time (s)	veh.	cost
LR1_4_1	**40**	**10639.75**	TS	10652.59	40.0	**10639.75**	**40**	351	**40**	**10639.75**
LR1_4_2	31	10533.33	SAM	10125.79	31.0	**10015.85**	**31**	554	**31**	**10015.85**
LR1_4_3	24	8831.1	SAM	8846.24	23.3	8908.01	23	613	**23**	**8840.46**
LR1_4_4	17	5551.47	LL	6974.01	16.0	6814.84	16	575	**16**	**6744.33**
LR1_4_5	30	10233.59	TS	10606.32	29.1	**10599.54**	**29**	457	**29**	**10599.54**
LR1_4_6	**25**	**9456.68**	BH	9686.93	25.0	9573.68	25	554	25	9525.45
LR1_4_7	21	8012.3	SAM	8170.00	19.7	**8200.37**	**19**	610	**19**	**8200.37**
LR1_4_8	15	6320.03	SAM	6093.04	14.1	6044.40	14	568	**14**	**5946.44**
LR1_4_9	25	10313.6	SAM	9908.16	24.7	**9886.14**	**24**	480	**24**	**9886.14**
LR1_4_10	22	8249.87	SAM	8233.16	21.0	8145.03	21	516	**21**	**8016.62**
LR2_4_1	**8**	**9726.88**	BH	10243.45	8.0	9786.02	8	467	8	9758.46
LR2_4_2	8	7971.09	SAM	9995.30	7.0	9717.03	7	761	**7**	**9496.64**
LR2_4_3	6	9794.4	SAM	8586.52	6.0	**8116.53**	**6**	1451	**6**	**8116.53**
LR2_4_4	5	5116.24	LL	6948.40	4.0	6695.51	4	3409	**4**	**6649.78**
LR2_4_5	7	9314.23	SAM	8893.25	7.0	8642.63	7	1096	**7**	**8574.84**
LR2_4_6	6	9439.98	SAM	8156.35	6.0	8089.75	6	1236	**6**	**7995.06**
LR2_4_7	5	7935.54	SAM	7126.64	5.0	**6928.61**	**5**	2019	**5**	**6928.61**
LR2_4_8	4	6043.41	LL	5591.83	4.0	**5447.40**	**4**	4603	**4**	**5447.40**
LR2_4_9	6	8552.29	SAM	8613.50	6.0	8135.86	6	780	**6**	**8043.20**
LR2_4_10	6	7449.9	TS	8008.78	5.2	**7904.77**	**5**	1385	**5**	**7904.77**
LC1_4_1	**40**	**7152.06**	SAM	7152.06	40.0	**7152.06**	**40**	585	**40**	**7152.06**
LC1_4_2	39	7326.93	BH	7395.61	38.9	**8012.43**	**38**	597	**38**	**8012.43**
LC1_4_3	35	7896.36	SAM	8538.36	33.1	**8308.94**	**33**	628	**33**	**8308.94**
LC1_4_4	**30**	**6451.68**	LL	7013.38	30.7	7021.92	30	558	30	6878.00
LC1_4_5	**40**	**7150**	SAM	7150.00	40.0	**7150.00**	**40**	508	**40**	**7150.00**
LC1_4_6	**40**	**7154.02**	LL	7154.02	40.0	**7154.02**	**40**	520	**40**	**7154.02**
LC1_4_7	**40**	**7149.43**	SAM	7149.43	40.0	**7149.43**	**40**	529	**40**	**7149.43**
LC1_4_8	**39**	**7111.16**	LL	7111.86	39.0	**7111.16**	**39**	542	**39**	**7111.16**
LC1_4_9	36	7539.92	SAM	7471.34	36.1	7458.43	36	462	**36**	**7452.21**
LC1_4_10	36	7181.05	TS	7278.25	35.8	7474.07	35	501	**35**	**7387.13**
LC2_4_1	**12**	**4116.33**	LL	4116.33	12.0	**4116.33**	**12**	319	**12**	**4116.33**
LC2_4_2	**12**	**4144.29**	SAM	4145.71	12.0	4144.49	12	455	**12**	**4144.29**
LC2_4_3	12	4624.76	SAM	4533.47	12.0	4483.34	12	681	**12**	**4431.75**
LC2_4_4	**12**	**3743.95**	LL	4123.21	12.0	4081.93	12	1169	12	4038.00
LC2_4_5	**12**	**4030.63**	TS	4030.97	12.0	4030.64	12	366	**12**	**4030.63**
LC2_4_6	**12**	**3900.29**	SAM	3905.41	12.0	3902.25	12	475	**12**	**3900.29**
LC2_4_7	**12**	**3962.51**	BH	3976.03	12.0	3969.69	12	481	**12**	**3962.51**
LC2_4_8	**12**	**3844.45**	SAM	3879.38	12.0	3867.31	12	549	**12**	**3844.45**
LC2_4_9	12	4198.61	SAM	4229.42	12.0	4209.49	12	604	**12**	**4188.93**
LC2_4_10	**12**	**3828.44**	BH	3846.45	12.0	3839.11	12	811	**12**	**3828.44**
LRC1_4_1	37	8944.58	TS	9059.11	36.5	**9127.15**	**36**	498	**36**	**9127.15**
LRC1_4_2	31	8642.74	SAM	8189.18	32.0	8404.51	31	550	**31**	**8346.06**
LRC1_4_3	**25**	**7307.09**	BH	7413.29	25.7	7429.00	25	644	25	7387.40
LRC1_4_4	19	5944.14	TS	5918.81	19.0	5901.86	19	909	**19**	**5838.58**
LRC1_4_5	34	9133.11	SAM	8760.38	34.0	8715.74	34	487	**33**	**8773.75**
LRC1_4_6	31	8817.39	SAM	8236.27	31.2	8198.96	31	475	**31**	**8177.90**
LRC1_4_7	30	7869.45	BH	7969.23	29.8	**7992.08**	**29**	500	**29**	**7992.08**
LRC1_4_8	28	7887.67	SAM	7625.79	27.9	**7613.43**	**27**	510	**27**	**7613.43**
LRC1_4_9	27	8215.25	SAM	7942.38	26.8	**8013.48**	**26**	494	**26**	**8013.48**
LRC1_4_10	24	7404.91	SAM	7190.05	24.0	7103.78	24	503	**24**	**7065.73**
LRC2_4_1	13	6655.52	SAM	7750.57	12.0	**7471.01**	**12**	553	**12**	**7471.01**
LRC2_4_2	11	7467.34	SAM	6385.15	11.0	6332.52	11	1102	**11**	**6303.36**
LRC2_4_3	9	5480.25	TS	5485.05	9.0	5459.06	9	2126	**9**	**5438.20**
LRC2_4_4	6	4279.05	LL	5446.01	5.0	5405.16	5	4032	**5**	**5322.43**
LRC2_4_5	**11**	**6120.13**	BH	6147.77	11.0	6140.07	11	827	**11**	**6120.13**
LRC2_4_6	10	6002.63	SAM	6540.83	9.1	**6479.56**	**9**	757	**9**	**6479.56**
LRC2_4_7	9	5737.02	SAM	6497.14	8.0	**6361.26**	**8**	707	**8**	**6361.26**
LRC2_4_8	8	5364.31	SAM	6004.71	7.1	5968.27	7	834	**7**	**5928.93**
LRC2_4_9	7	6892.23	SAM	5469.65	7.0	5394.73	7	1275	**7**	**5303.53**
LRC2_4_10	7	5057.81	TS	6124.51	6.0	**5760.78**	**6**	1243	**6**	**5760.78**
Tot.	1183	421215		425816.87	1167.80	422201.17	1158	52850	1157	420395.99
Avg.								881		
< PB						40				41
<= PB						47				55
#B		19				25				55

Table 18: Results on 400-customer problems

	Best known			FULL					LNS best known	
	veh.	cost	References	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. time (s)	veh.	cost
LR1_6_1	**59**	**22838.3**	BVH	23070.74	59.0	22975.40	59	1443	59	22838.65
LR1_6_2	45	20985.7	BVH	20714.68	45.0	20614.87	45	1438	**45**	**20246.18**
LR1_6_3	37	18685.9	BVH	18619.94	37.0	18548.01	37	1620	**37**	**18073.14**
LR1_6_4	28	13945.59	TS	13677.43	28.0	13604.92	28	2119	**28**	**13269.71**
LR1_6_5	39	22985.63	SAM	21983.13	39.0	**22562.81**	**38**	1105	**38**	**22562.81**
LR1_6_6	33	21427.75	SAM	20373.88	33.0	20060.42	33	1299	**32**	**20641.02**
LR1_6_7	27	17070.51	SAM	16615.48	26.6	16746.97	26	1476	**25**	**17162.90**
LR1_6_8	20	12669.88	SAM	12412.57	19.0	12302.45	19	1916	**19**	**11957.59**
LR1_6_9	34	21273.3	BVH	20917.36	33.2	20765.52	33	1059	**32**	**21423.05**
LR1_6_10	28	19337.5	SAM	18400.79	28.0	18233.75	28	989	**27**	**18723.13**
LR2_6_1	12	18840.8	BVH	22245.55	11.0	22049.96	11	1245	**11**	**21945.30**
LR2_6_2	11	17452.75	TS	20038.78	10.0	**19666.59**	**10**	2089	**10**	**19666.59**
LR2_6_3	9	17598.73	SAM	16161.38	8.0	15897.51	8	3729	**8**	**15609.96**
LR2_6_4	7	11771.45	TS	11627.85	6.0	10916.25	6	12849	**6**	**10819.45**
LR2_6_5	10	19347.2	SAM	20529.74	9.0	20079.56	9	1300	**9**	**19567.41**
LR2_6_6	9	19889.05	SAM	18788.90	8.0	17599.80	8	2238	**8**	**17262.96**
LR2_6_7	7	16262	BVH	16052.41	6.0	15877.37	6	6915	**6**	**15812.42**
LR2_6_8	6	11652.95	TS	11175.02	5.0	11026.09	5	10329	**5**	**10950.90**
LR2_6_9	9	18853.4	BVH	19465.02	8.0	19180.31	8	2123	**8**	**18799.36**
LR2_6_10	7	18449.18	SAM	17599.63	7.0	17261.53	7	1928	**7**	**17034.63**
LC1_6_1	**60**	**14095.6**	LL	14095.64	60.0	14095.64	60	1453	60	14095.64
LC1_6_2	**58**	**14379.5**	BVH	14383.04	58.0	14380.37	58	1440	58	14379.53
LC1_6_3	51	14569.3	BVH	14676.36	50.8	15028.86	50	1153	**50**	**14683.43**
LC1_6_4	48	13567.51	LL	13806.44	49.0	13750.06	49	1066	**47**	**13648.03**
LC1_6_5	**60**	**14086.3**	LL	14086.30	60.0	**14086.30**	**60**	1201	**60**	**14086.30**
LC1_6_6	**60**	**14090.79**	SAM	14090.79	60.0	**14090.79**	**60**	1198	**60**	**14090.79**
LC1_6_7	**60**	**14083.76**	SAM	14083.76	60.0	**14083.76**	**60**	1203	**60**	**14083.76**
LC1_6_8	**59**	**14554.27**	SAM	14557.89	59.0	14554.81	59	1263	**59**	**14554.27**
LC1_6_9	55	14626.25	TS	14676.34	56.0	14596.57	56	1261	**54**	**14706.12**
LC1_6_10	54	14627.2	TS	14918.57	55.6	14711.59	55	1329	**53**	**14879.30**
LC2_6_1	**19**	**7977.98**	SAM	7977.98	19.0	**7977.98**	**19**	1137	**19**	**7977.98**
LC2_6_2	19	8253.67	SAM	10612.70	18.0	10384.03	18	1277	**18**	**10277.23**
LC2_6_3	18	7436.5	BVH	7781.67	17.8	9007.34	17	2033	**17**	**8728.30**
LC2_6_4	18	8200.89	TS	8279.98	17.2	8281.94	17	2303	**17**	**8041.97**
LC2_6_5	**19**	**8047.37**	BVH	8068.59	19.0	8061.74	19	1268	**19**	**8047.37**
LC2_6_6	19	8169.95	TS	8149.37	19.0	8129.87	19	1016	**19**	**8094.11**
LC2_6_7	19	8038.56	BVH	8108.38	19.0	8086.65	19	1133	**19**	**7998.18**
LC2_6_8	18	7808.16	SAM	7632.38	18.0	7616.85	18	1067	**18**	**7579.93**
LC2_6_9	19	8134.25	SAM	8173.11	19.0	8160.19	19	1225	**18**	**9501.00**
LC2_6_10	18	7555.35	TS	7529.02	18.0	7511.89	18	1775	**17**	**8019.94**
LRC1_6_1	53	17930	BVH	18017.12	53.0	17965.79	53	1342	**53**	**17924.88**
LRC1_6_2	45	16040.3	BVH	16090.72	44.8	**16302.54**	**44**	1389	**44**	**16302.54**
LRC1_6_3	36	14407.6	BVH	14395.28	36.0	14310.59	36	1725	**36**	**14060.31**
LRC1_6_4	25	11308.6	BVH	11260.62	25.0	11097.51	25	2496	**25**	**10950.52**
LRC1_6_5	47	16803.9	BVH	16837.12	47.8	16831.90	47	1256	**47**	**16742.55**
LRC1_6_6	44	18205.25	SAM	17059.61	45.0	16994.01	45	1175	**44**	**16894.37**
LRC1_6_7	39	16407.68	SAM	15582.48	39.6	15565.62	39	1135	**39**	**15394.87**
LRC1_6_8	36	15352.6	BVH	15346.86	36.0	15174.29	36	1099	**36**	**15154.79**
LRC1_6_9	36	15751.84	SAM	15092.82	36.2	15000.49	36	1141	**35**	**15134.24**
LRC1_6_10	31	14304.37	SAM	14036.50	32.0	13940.77	32	1058	**31**	**13925.51**
LRC2_6_1	17	13111.6	BVH	14989.05	16.0	14844.71	16	1194	**16**	**14817.72**
LRC2_6_2	15	11463	BVH	12856.00	14.0	12801.40	14	2106	**14**	**12758.77**
LRC2_6_3	11	15167.3	BVH	12413.60	10.6	**12812.67**	**10**	4830	**10**	**12812.67**
LRC2_6_4	8	12512.5	BVH	10461.14	7.4	**10574.87**	**7**	13452	**7**	**10574.87**
LRC2_6_5	14	15576.76	SAM	13287.40	14.0	13216.21	14	1827	**14**	**13009.52**
LRC2_6_6	13	12655.11	SAM	12717.44	13.0	12709.04	13	1826	**13**	**12643.98**
LRC2_6_7	11	13996.73	SAM	12109.64	11.0	12070.35	11	1397	**11**	**12007.65**
LRC2_6_8	11	14572.07	SAM	12681.15	10.0	12565.94	10	2341	**10**	**12163.43**
LRC2_6_9	10	12262.51	TS	14236.58	9.0	13966.61	9	2094	**9**	**13768.01**
LRC2_6_10	9	12379.46	TS	12300.10	8.0	12129.35	8	2340	**8**	**12016.94**
Tot.	1699	873850		867929.80	1686.60	863441.95	1679	133234	1664	860898.44
Avg.								2221		
< PB						41				51
<= PB						45				57
#B		9				9				57

Table 19: Results on 600-customer problems

	Best known			FULL					LNS best known	
	veh.	cost	References	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. time (s)	veh.	cost
LR181	80	39374.4	LL	39847.80	80.0	39719.88	80	2867	**80**	**39315.92**
LR182	59	36122.5	BVH	35197.46	59.0	34746.99	59	2719	**59**	**34370.37**
LR183	45	31763	BVH	30506.10	44.0	30301.99	44	2984	**44**	**29718.09**
LR184	26	23454.57	SAM	21738.05	25.6	21900.66	25	3458	**25**	**21197.65**
LR185	52	39743.88	SAM	37834.13	52.4	37856.78	52	2051	**50**	**39046.06**
LR186	42	35011.85	SAM	33815.72	42.6	34315.99	42	2250	**42**	**33659.50**
LR187	34	28551.92	SAM	27347.55	32.8	28327.14	32	2720	**32**	**27294.19**
LR188	24	21891.97	SAM	20182.46	21.2	20256.27	21	2982	**21**	**19570.21**
LR189	44	36550.5	SAM	35693.92	43.0	35531.29	43	1890	**42**	**36126.69**
LR1810	34	31443.25	SAM	29741.89	33.6	29587.53	33	1891	**32**	**30200.86**
LR281	16	29961.22	SAM	34422.50	15.0	34124.11	15	2009	**15**	**33816.90**
LR282	13	37565.81	SAM	30839.74	12.8	33326.43	12	4507	**12**	**32575.97**
LR283	11	30046.47	SAM	26211.39	10.0	25446.52	10	8134	**10**	**25310.53**
LR284	8	24925.57	SAM	20085.04	7.0	**19506.42**	**7**	24419	**7**	**19506.42**
LR285	12	34256.18	SAM	34919.19	12.0	33961.98	12	2515	**12**	**32634.29**
LR286	10	30688.6	SAM	29070.99	10.0	28629.45	10	5827	**10**	**27870.80**
LR287	9	28524.9	BVH	25809.90	8.0	**25077.85**	**8**	7397	**8**	**25077.85**
LR288	7	19878.42	TS	18168.34	6.0	17800.02	6	29265	**5**	**19256.79**
LR289	11	34700.25	SAM	30325.20	10.8	31891.23	10	3025	**10**	**30791.77**
LR2810	10	31906.16	SAM	29604.30	9.0	28941.03	9	3425	**9**	**28265.24**
LC181	**80**	**25184.38**	SAM	25184.38	80.0	**25184.38**	**80**	2663	**80**	**25184.38**
LC182	**78**	**26056.2**	BVH	26186.79	78.0	26131.65	78	2712	78	26062.17
LC183	66	26700.6	BVH	26135.96	66.8	26308.88	66	2591	**65**	**25918.45**
LC184	61	23427.2	BVH	23880.34	62.4	23786.46	62	1892	**60**	**22970.88**
LC185	**80**	**25211.22**	SAM	25211.22	80.0	**25211.22**	**80**	2207	**80**	**25211.22**
LC186	**80**	**25164.25**	SAM	25164.25	80.0	**25164.25**	**80**	2210	**80**	**25164.25**
LC187	**80**	**25158.38**	SAM	25158.38	80.0	**25158.38**	**80**	2249	**80**	**25158.38**
LC188	78	25427.1	BVH	25262.20	79.0	25255.06	79	2187	**78**	**25348.45**
LC189	74	25536	BVH	26352.66	75.4	26363.13	74	2488	**73**	**25541.94**
LC1810	72	26364.93	TS	26896.75	75.0	26522.79	74	2394	**71**	**25712.12**
LC281	**24**	**11687.06**	SAM	11687.06	24.0	**11687.06**	**24**	1030	**24**	**11687.06**
LC282	25	12575	BVH	12634.54	25.0	12614.42	25	2462	**24**	**14358.92**
LC283	25	12500.5	BVH	13687.38	24.0	13551.68	24	2010	**24**	**13198.29**
LC284	24	13438.1	TS	12662.06	24.0	12593.32	24	3046	**23**	**13376.82**
LC285	**25**	**12298.9**	BVH	12357.15	25.0	12350.55	25	1237	25	12329.80
LC286	25	12064.8	BVH	12112.84	25.0	12090.57	25	1713	**24**	**12702.87**
LC287	25	11899.18	TS	11895.72	25.0	11878.10	25	1360	**25**	**11855.86**
LC288	24	11724.46	TS	11649.71	24.0	11592.23	24	1520	**24**	**11482.88**
LC289	24	11700.86	TS	11685.81	24.0	11673.27	24	1862	**24**	**11629.61**
LC2810	24	12139.06	TS	11693.40	24.0	11615.76	24	1874	**24**	**11578.58**
LRC181	67	32587.9	BVH	32275.83	67.6	**32268.95**	**67**	2206	**67**	**32268.95**
LRC182	**56**	**28843.1**	BVH	28306.81	58.4	28180.05	58	2515	57	28395.39
LRC183	**49**	**24933.9**	BVH	24672.74	51.0	24628.67	51	3207	50	24354.36
LRC184	35	18768.4	BVH	18696.22	35.0	18666.34	35	4276	**35**	**18241.91**
LRC185	**60**	**32578.04**	SAM	31439.49	63.2	31121.74	63	2218	61	30995.48
LRC186	**56**	**29971.97**	SAM	29037.55	59.8	28934.95	59	2135	58	28568.61
LRC187	**53**	**29948.45**	SAM	28696.11	55.8	28543.20	55	1944	54	28164.41
LRC188	49	28160.88	SAM	26889.40	50.8	26971.48	50	2105	**49**	**26150.65**
LRC189	47	26668.91	SAM	25538.12	48.6	25578.39	48	2016	**47**	**24930.70**
LRC1810	43	25787.27	SAM	24424.49	44.2	24156.12	44	2004	**42**	**24271.52**
LRC281	21	21486.1	LL	21905.03	20.8	23476.51	20	2217	**20**	**23289.40**
LRC282	19	19127.96	SAM	20056.42	19.2	19930.17	19	3522	**18**	**21786.62**
LRC283	17	18842.56	TS	16423.77	16.4	16846.85	16	6751	**16**	**16586.31**
LRC284	13	17693.9	BVH	14406.39	12.0	**14122.05**	**12**	19037	**12**	**14122.05**
LRC285	18	21626.63	TS	20541.12	18.0	20474.88	18	2725	**18**	**20292.92**
LRC286	16	25106.28	SAM	21271.46	16.0	21209.60	16	2792	**16**	**21088.57**
LRC287	15	23808.4	SAM	20402.90	15.0	19764.32	15	3187	**15**	**19695.96**
LRC288	13	24260	SAM	19670.06	13.0	19423.27	13	3722	**13**	**19009.33**
LRC289	13	19514	BVH	19548.71	12.0	19267.46	12	3702	**12**	**19003.68**
LRC2810	12	19865.4	BVH	19257.95	10.8	20530.09	10	4736	**10**	**19766.78**
Tot.	2213	1492200		1432320.80	2223.00	1432077.81	2208	235063	2181	1423062.65
Avg.								3918		
< PB						37				48
<= PB						42				53
#B		12				9				53

Table 20: Results on 800-customer problems

	Best known			FULL, Both IA = 0.01					LNS best known	
	veh.	cost	References	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. time (s)	veh.	cost
LR1101	100	57977	BVH	57172.54	100.0	57016.58	100	4576	**100**	**56903.88**
LR1102	80	52361.61	SAM	49937.45	80.0	49765.70	80	4495	**80**	**49652.10**
LR1103	54	44890.55	SAM	42886.53	54.0	42681.33	54	4473	**54**	**42124.44**
LR1104	31	32336.04	SAM	31450.33	29.0	**32133.36**	**28**	4522	**28**	**32133.36**
LR1105	64	58260.68	SAM	58138.72	61.6	**59135.86**	**61**	3474	**61**	**59135.86**
LR1106	51	49697.85	SAM	47333.63	51.8	**48637.63**	**50**	3673	**50**	**48637.63**
LR1107	39	39861.97	SAM	38315.35	38.2	**38936.54**	**37**	3598	**37**	**38936.54**
LR1108	29	31515.87	SAM	29674.35	26.4	**29452.32**	**26**	4892	**26**	**29452.32**
LR1109	52	52282.36	SAM	51412.70	51.0	**52223.15**	**50**	3126	**50**	**52223.15**
LR11010	42	45710.21	SAM	45873.80	41.0	**46218.35**	**40**	2841	**40**	**46218.35**
LR2101	19	45835.55	SAM	47201.18	19.0	45493.36	19	3158	**19**	**45422.58**
LR2102	16	48817.75	SAM	51094.71	15.4	50925.97	15	5324	**15**	**47824.44**
LR2103	13	43094.14	SAM	38654.94	12.0	37778.15	12	12055	**11**	**39894.32**
LR2104	10	32993.09	SAM	28821.03	8.6	29783.60	8	26496	**8**	**28314.95**
LR2105	15	56010.62	SAM	53453.03	14.8	55497.90	14	4244	**14**	**53209.98**
LR2106	13	48225.07	SAM	46388.49	12.4	46145.75	12	6565	**12**	**43792.11**
LR2107	11	38336.76	SAM	36506.87	9.6	38322.91	9	14455	**9**	**36728.20**
LR2108	8	32493.7	SAM	27137.04	7.0	26631.41	7	26592	**7**	**26278.09**
LR2109	14	55587.14	SAM	52093.74	13.0	50990.04	13	3844	**13**	**48447.49**
LR21010	12	47678.69	SAM	44815.46	11.6	46117.94	11	5945	**11**	**44155.66**
LC1101	**100**	**42488.66**	SAM	42488.66	100.0	**42488.66**	**100**	4025	**100**	**42488.66**
LC1102	96	43437.2	BVH	43417.56	95.8	**43870.19**	**95**	4008	**95**	**43870.19**
LC1103	85	42483.61	SAM	42589.34	82.6	**42631.11**	**82**	4123	**82**	**42631.11**
LC1104	76	39613.83	SAM	38950.40	74.8	**39443.00**	**74**	3617	**74**	**39443.00**
LC1105	**100**	**42477.4**	SAM	42477.41	100.0	42477.41	100	3603	100	42477.41
LC1106	**101**	**42838.39**	SAM	42838.39	101.0	**42838.39**	**101**	3714	**101**	**42838.39**
LC1107	**100**	**42854.99**	TS	42855.17	100.0	**42854.99**	**100**	3556	**100**	**42854.99**
LC1108	99	42711.7	BVH	42964.24	98.0	42954.34	98	3637	**98**	**42951.56**
LC1109	93	42899.1	BVH	42614.87	92.2	**42391.98**	**92**	3508	**92**	**42391.98**
LC11010	91	42243.4	TS	42715.95	90.2	**42435.16**	**90**	3582	**90**	**42435.16**
LC2101	**30**	**16879.24**	TS	16879.24	30.0	**16879.24**	**30**	1502	**30**	**16879.24**
LC2102	32	17598.6	BVH	19210.16	31.4	19116.33	31	2171	**31**	**18980.98**
LC2103	30	19198.95	SAM	17503.99	30.8	17940.74	30	3651	**30**	**17772.49**
LC2104	30	17726	LL	19076.31	30.2	18418.52	30	4120	**29**	**18089.93**
LC2105	31	17466.42	TS	17149.07	31.0	**17137.53**	**31**	2561	**31**	**17137.53**
LC2106	31	17352.7	TS	18276.39	31.0	17217.15	31	2012	**31**	**17198.01**
LC2107	32	18131.36	TS	19306.15	32.0	17721.20	32	2796	**31**	**19117.67**
LC2108	30	17974.2	SAM	17266.57	30.0	17035.24	30	2745	**30**	**17018.63**
LC2109	31	17769.6	BVH	17825.02	31.2	17667.44	31	2809	**31**	**17565.95**
LC21010	30	18249.85	SAM	18342.21	30.2	17266.19	30	3297	**29**	**17425.55**
LRC1101	**84**	**49315.3**	BVH	48997.27	85.4	48934.66	85	3638	85	48702.83
LRC1102	73	45679.5	BVH	45351.71	73.0	45272.96	73	3966	**73**	**45135.70**
LRC1103	55	36570.5	BVH	35393.15	55.4	**35475.72**	**55**	4397	**55**	**35475.72**
LRC1104	41	28979.2	BVH	28013.33	40.2	27930.03	40	6042	**40**	**27747.04**
LRC1105	76	51455.4	BVH	50012.71	76.2	**49816.18**	**76**	3372	**76**	**49816.18**
LRC1106	69	47014.55	SAM	44308.41	70.2	**44469.08**	**69**	3132	**69**	**44469.08**
LRC1107	65	43321.51	SAM	41395.55	65.2	**41413.16**	**64**	3047	**64**	**41413.16**
LRC1108	60	42968.34	SAM	40946.68	61.0	**40590.17**	**60**	3017	**60**	**40590.17**
LRC1109	57	42549.12	SAM	39708.07	58.0	**39587.85**	**57**	2837	**57**	**39587.85**
LRC11010	**51**	**38274.02**	SAM	36184.43	52.2	36195.00	52	2930	52	36195.00
LRC2101	23	36894.98	SAM	32969.29	23.2	**35073.70**	**22**	2864	**22**	**35073.70**
LRC2102	22	28019.7	LL	29945.79	22.2	31054.84	21	4749	**21**	**30932.74**
LRC2103	19	30226.39	SAM	27201.83	17.8	28662.28	17	9528	**16**	**28403.51**
LRC2104	14	25836.7	BVH	22976.06	12.8	23611.31	12	28075	**12**	**23083.20**
LRC2105	19	39344.9	SAM	31946.46	18.8	**34713.96**	**18**	3945	**18**	**34713.96**
LRC2106	18	29947.9	SAM	30362.74	18.0	29577.50	18	2356	**17**	**31485.26**
LRC2107	18	31633.3	BVH	29915.31	17.2	29822.82	17	4432	**17**	**29639.63**
LRC21010	13	31361.45	SAM	30293.97	12.2	30160.05	12	5729	**12**	**29402.90**
Tot.	2698	2195755		2129031.74	2677.80	2137033.93	2652	311441	2646	2122921.51
Avg.								5370		
< PB						50			51	
<= PB						54			55	
#B		7				25			55	

Table 21: Results on 1000-customer problems

Chapter 5

A unified heuristic for a large class of vehicle routing problems with backhauls

A Unified Heuristic for a Large Class of Vehicle Routing Problems with Backhauls

Stefan Ropke and David Pisinger *

Abstract

The Vehicle Routing Problem with Backhauls is a generalization of the ordinary capacitated vehicle routing problem where goods are delivered from the depot to the linehaul customers, and additional goods are brought back to the depot from the backhaul customers. Numerous ways of modeling the backhaul constraints have been proposed in the literature, each imposing different restrictions on the handling of backhaul customers. A survey of these models is presented, and a unified model is developed that is capable of handling most variants of the problem from the literature. The unified model can be seen as a Rich Pickup and Delivery Problem with Time Windows, which can be solved through an improved version of the large neighborhood search heuristic proposed by Ropke (2003). The results obtained in this way are comparable to or improve on similar results found by state of the art heuristics for the various variants of the problem. The heuristic has been tested on 338 problems from the literature and it has improved the best known solution for 227 of these. An additional benefit of the unified modeling and solution method is that it allows the dispatcher to mix various variants of the Vehicle Routing Problem with Backhauls for the individual customers or vehicles.

Keywords: metaheuristics, vehicle routing problems, large neighborhood search

1 Introduction

In the classical *Capacitated Vehicle Routing Problem* (CVRP) we have to deliver goods from a depot to a set of customers, using a set of identical vehicles. Each customer demands a certain quantity of goods and the vehicles have a limited capacity. Our task is to construct routes starting and ending at the depot that minimize the total travel distance and that obey the capacity of the vehicles.

The problems that need to be solved in real life situations are usually much more complicated. One complication that arises in practice is that goods not only need to be brought from the depot to the customers, but also must be picked up at a number of customers and brought back to the depot. A simple way of handling such problems is to solve two independent CVRPs. One for the delivery (*linehaul*) customers and one for the pickup (*backhaul*) customers, such that some vehicles would be designated to linehaul customers and others to backhaul customers. This approach is not likely to create high quality solutions though — it seems more profitable to serve both pickup and delivery customers using the same vehicles. The *Vehicle Routing Problem with Backhauls* (VRPB) models problems with both pickup and delivery customers in the same route.

Applications of VRPB can be found in the distribution of groceries. Groceries are delivered to supermarkets and grocery stores from a central distribution center and groceries are picked up at production sites and brought to the distribution center. Another application is the handling of returnable bottles, where full bottles are brought to customers and empty bottles are brought back to breweries to be recycled. Such applications are likely to become more common in the future due to the increased awareness of environmental issues. It is important to develop fast and robust algorithms for real-life transportation problems, which are able to handle various side constraints that appear in practice.

*DIKU - Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark. E-mail: {sropke, pisinger}@diku.dk

The general trend in the transportation sector is that transportation companies are merging to larger units which can provide a large number of delivery services. In order to get the most possible benefit from the vehicle fleet, it can be attractive to service conceptually different transportation tasks by the same fleet, thus models are needed that can handle all additional constraints associated with a transportation task. Cordeau et al. [6] for example provide a unified approach for several Vehicle Routing Problems with Time Windows. The present paper considerably extends the expressibility of the model, by also allowing pickup and delivery requests, precedence constraints, etc. This allows us to formulate the six most common variants of vehicle routing problems with backhauls within the framework, and to find high quality heuristic solutions that are comparable to or improve on similar results for specialized algorithms.

The underlying problem of all of the problems we consider is the *Pickup and Delivery Problem with Time Windows* (PDPTW), which we will describe in Section 2. A survey of the six most common variants of vehicle routing problems with backhauls — and additional, less frequently used models — is given in Section 3. The subsequent sections present the heuristic algorithm proposed in this paper, which is outlined in Figure 1. Some of the problem types we wish to solve are illustrated at the top of the figure. To solve an instance of one of these problem types, we transform it to an instance of the *Rich Pickup and Delivery Problem with Time Windows*, as illustrated by the arrows from the top row to the next row. Transformations are discussed in Section 4. The PDPTW instance is solved by a heuristic which will be presented in Section 5; this produces a PDPTW solution that finally is interpreted as a solution to the original problem. This solution framework has been tested on 338 benchmarks problems proposed in the literature. The results of this computational test are reported in Section 6. The paper is finally concluded in Section 7.

2 The Pickup and delivery problem with time windows (PDPTW)

Before starting to discuss the various variants of the VRPB we introduce the *Rich Pickup and Delivery Problem with Time Windows* (Rich PDPTW). All considered variants of the VRPB can be seen as extensions of the PDPTW. IP models of the PDPTW can be found in Desaulniers et al. [8] and Sigurd et al. [34], for our purpose we will only give a verbal description of our problem which differs slightly from the problems in the afore-mentioned papers.

In the Rich PDPTW we have *n requests* and *m vehicles*. A *request* $i \in \{1,...,n\}$ consists of picking up a quantity l_i of goods at one location and delivering it to another location. With each request is associated a *pickup time window*, a *delivery time window*, and two *service times* s_i^p and s_i^d indicating how long the pickup and delivery operations take to perform. A vehicle is allowed to arrive at a location before the start of the time window, in which case it will have to wait before starting the corresponding operation. A vehicle may never arrive at a location after the end of the time window. Each request furthermore has an associated *pickup precedence number*, and a *delivery precedence number*. Each vehicle must visit the locations in nondecreasing order of precedence number (see e.g. Sigurd et al. [34] for various applications of precedence constraints).

Each request i can only be served by a vehicle $k \in F_i$, where F_i is the set of feasible vehicles corresponding to request i. Each vehicle $k \in \{1,\dots,m\}$ has an associated *capacity* C_k, a *start time* b_k and *end time* e_k, and an associated *start terminal* B_k and *end terminal* E_k where it starts and ends its duty respectively. The vehicle must leave its start terminal at time b_k even though this might introduce waiting time at the first customer visited. The vehicle must return to the end terminal at time e_k or before.

The problem can be defined on a directed graph where the locations are represented by a set of *nodes* $V = \{1,...,2n+2m\}$, and for each *edge* (i,j) we have an associated *distance* d_{ij} and *travel time* t_{ij}, where we assume that travel times satisfy the triangle inequality while the only assumption on the distances is that they must be non-negative. The locations will often be referred to as *visits*.

The task is to construct a set of valid routes for a limited number of vehicles such that an associated *objective function* is minimized. The objective function is a weighted sum of 1) the sum of the distance traveled by the vehicles. 2) the number of requests not assigned to a vehicle. The two terms are weighted by the coefficients α and β. Notice that this objective function does not necessarily assign all requests to a vehicle. Requests not assigned to a vehicle are placed in a virtual *request bank*, which in a real world situation must be handled by a human dispatcher.

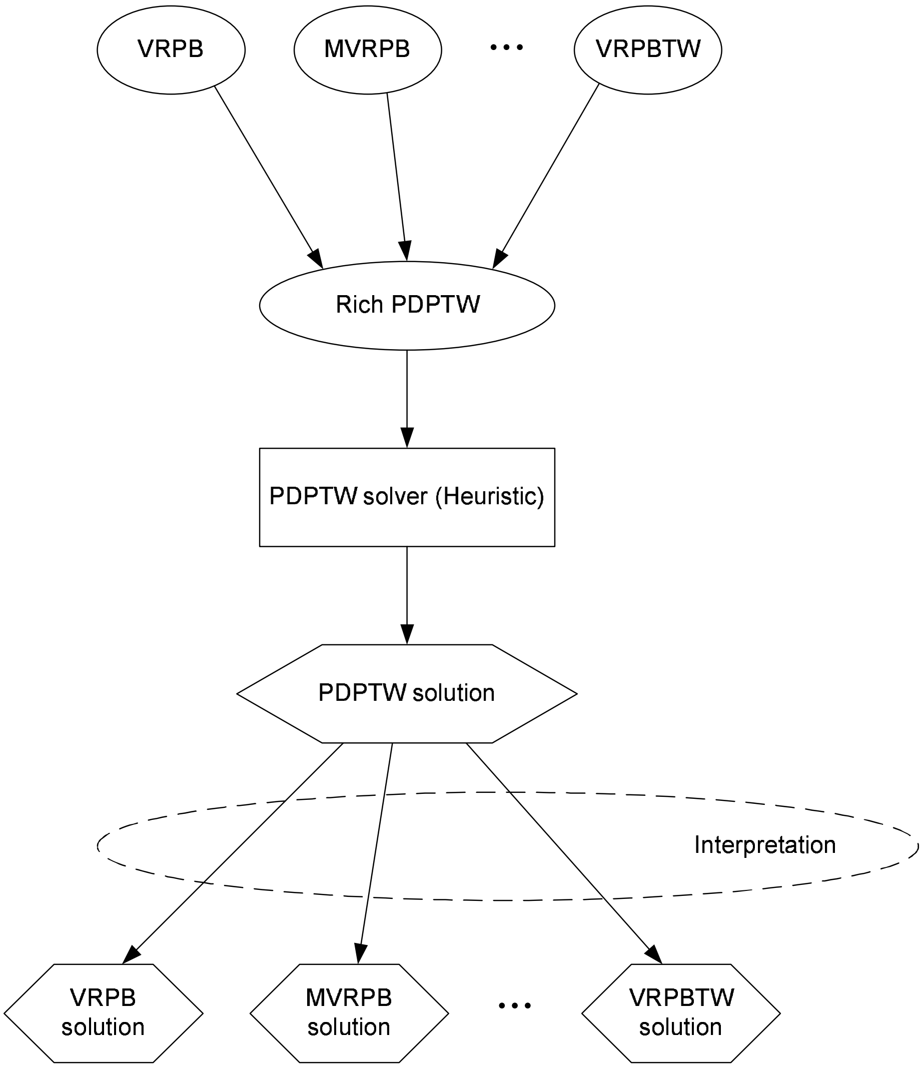


Figure 1: Solution framework: As described in Section 3 the algorithm accepts as input variants of the Vehicle Routing Problem with Backhauls, including: (VRPB), (MVRPB), (MDMVRPB), (VRPBTW), (MVRPBTW) and (VRPSDP). All of the problems are transformed to a Rich Pickup and Delivery Problem with Time Windows, which is solved heuristically through a Large Neighborhood Search algorithm. The last step of the algorithm transforms the obtained solution back to the original problem. The framework is not limited to backhaul models, but can be used to solve other types of vehicle routing problems, such as the vehicle routing problem with time windows or the capacitated vehicle routing problem.

Hence, normally a high value is assigned to the coefficient β to stimulate that as many requests as possible are to be serviced. In the experiments performed in this paper, β was chosen sufficiently high to avoid situations were some requests where left in the request bank upon termination.

3 Overview of vehicle routing problems with backhauls

This section gives an overview of the vehicle routing problems with backhauls proposed in the literature. We restrict ourselves to multi-vehicle problems. Single-vehicle problems have been studied by for example Gendreau et al. [14], Ghaziri and Osman [15] and Süral and Bookbinder [36].

3.1 The Vehicle Routing Problem with Backhauls (VRPB)

In the *vehicle routing problem with backhauls* (VRPB) we wish to minimize the total traveled distance and we are allowed to serve linehaul and backhaul customers on the same routes subject to the following limitations.

(A) If a route contains both linehaul and backhaul customers then the backhaul customers must be served after the linehaul customers.
(B) A route is not allowed to consist entirely of backhaul customers.
(C) The capacity of the vehicle should be obeyed, that is, neither the sum of the demands of the linehaul customers nor the sum of the demands of the backhaul customers served by a vehicle may exceed the vehicle capacity.
(D) The number of vehicles to use is given in advance. This means that even if it is possible to find better solutions using fewer or more vehicles, we must report the best solution we can find that uses the specified number of vehicles.
(E) All customers are serviced from a single depot.
(F) All vehicles have the same capacity.

Constraint (A) might seem artificial but it is justified by the fact that many vehicles are rear-loaded. This makes it problematic to try to load the vehicle with goods heading for the depot before we have delivered all goods to the customers as the pickup goods might block access to the delivery goods. The constraint is also justified by the fact that the linehaul customers frequently prefer early deliveries while backhaul customers prefer late pickups.

A recent survey of the VRPB was presented by Toth and Vigo [42]. Exact methods for the VRPB are proposed by Mingozzi et al. [26] and Toth and Vigo [41]. Heuristics have been developed by Anily [3], Casco et al. [5], Crispim and Brandao [7], Goetschalckx and Jacobs-Blecha [16], [22] and Toth and Vigo [40].

3.2 The Mixed Vehicle Routing Problem with Backhauls (MVRPB)

The *Mixed Vehicle Routing Problem with Backhauls* (MVRPB) is derived from the VRPB by relaxing limitations (A), (B) and (D). That is, we can mix linehaul and backhaul customers freely within a route and we are free to use as many vehicles as we want. We still have to obey the capacity limit of the vehicles. The capacity check is slightly more complicated in the MVRPB problem as the vehicle load fluctuates during the route. Furthermore, some MVRPB also have a duration limit that implies that routes should be completed within a certain time frame; for such problems the travel time between customers and the service time at the customers is given.

The name *Vehicle Routing Problem with Pickups and Deliveries* (VRPPD) is sometimes used instead of MVRPB. Heuristics for this problem are presented by Halse [19], Nagy and Salhi [27], [32] and Wade and Salhi [43], [44].

3.3 The Multiple Depot Mixed Vehicle Routing Problem with Backhauls (MDMVRPB)

The *Multiple Depot Mixed Vehicle Routing Problem with Backhauls* (MDMVRPB) is a generalization of the MVRPB. In the MDVRPB limitation (E) is relaxed such that we instead of just considering a single depot are faced with problems where several depots are present. At each depot a limited fleet of vehicles is available, and a

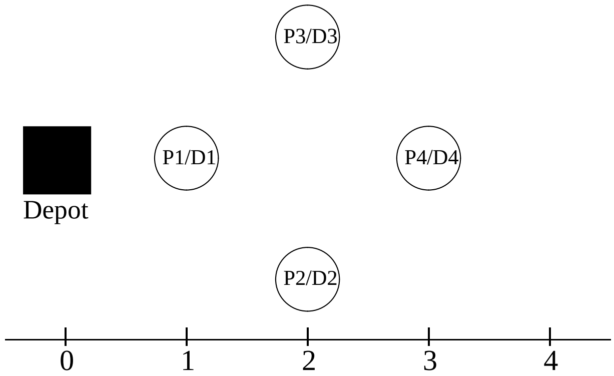


Figure 2: An example showing that simultaneous pickup and delivery at customers may increase the overall route lengths. The four customers have pickup/delivery requests of 2/2, 1/2, 1/2, 2/0 respectively. The vehicle has a capacity C of 6 units, and normal Euclidean distances are used. In a MVRPB setting, the shortest route is D1, P2/D2, P4/D4, P3/D3, P1 of total length 7.66. If simultaneous pickup and deliveries are demanded, the shortest route becomes P3/P3, P2/D2, P4/D4, P1/D1 of total length 8.65.

vehicle should start and end its duty at the same depot. Heuristics for the problem are proposed by Nagy and Salhi [27], [32]. They denoted the problem the *Multi Depot Vehicle Routing Problem with Pickup and Deliveries*.

3.4 The Vehicle Routing Problem with Backhauls and Time Windows (VRPBTW)

The *Vehicle Routing Problem with Backhauls and Time Windows* (VRPBTW) extends VRPB by assigning a time window to each customer, by having travel times associated with each pair of locations, and by having service times associated with the customers. Visits at a customer should start within the time window. If the vehicle arrives too early at a customer it has to wait until the start of the time window. If the vehicle arrives too late the route is invalid. Limitations (B) and (D) from the VRPB are relaxed in the VRPBTW. The objective of VRPBTW is either to minimize the total traveled distance or to minimize the number of vehicles as the first priority and then minimize the total traveled distance as the second priority.

An exact algorithm for the VRPBTW based on column generation was proposed by Gelinas et al. [13], and heuristics were proposed by Duhamel et al. [12], Hasama et al. [20], Reimann et al. [30], Thangiah et al. [38] and Zhong and Cole [48].

3.5 The Mixed Vehicle Routing Problem with Backhauls and Time Windows (MVRPBTW)

The *Mixed Vehicle Routing Problem with Backhauls and Time Windows* (MVRPBTW) is derived from VRPBTW by relaxing limitation (A) saying that backhaul customers should be visited after linehaul customers. The objective that has been considered in the literature is to minimize the number of vehicles as the first priority and the distance traveled as the second priority. Two heuristics have been proposed in Kontoravdis and Bard [23] and Zhong and Cole [48].

3.6 The Vehicle Routing Problem with Simultaneous Deliveries and Pickups (VRPSDP)

In the *Vehicle Routing Problem with Simultaneous Deliveries and Pickups* (VRPSDP) a subset of the customers simultaneously demand goods from—and supply goods to—the depot, and thus both a delivery and a pickup should occur at these customers. The pickup and delivery should be performed simultaneously such that each customer is visited only once by a vehicle. Unloading is obviously done before loading at these customers. The simultaneous pickup and delivery operation decreases the customers' expenses or inconvenience associated with handling vehicles, but may result in longer routes as illustrated in Figure 2.

This problem was first introduced by Min [25] in the context of transportation material between public libraries and a library administration center (acting as a depot). Halse [19] presented exact and heuristic methods for the problem and Dethloff [9], [10] considered heuristic algorithms. Nagy and Salhi [32] used their MVRPB heuristic to solve the problem, but apparently the “simultaneous” constraint is not handled by the heuristic. This is discussed in further detail by Dethloff [10]. Two variants of the problem have been proposed recently. Nagy and Salhi [32] introduce a multi depot version of the problem, while Angelelli and Mansini [2] solve a version with time windows to optimality using column generation. The heuristic proposed in the present paper is not tested on the two last problem types although the underlying PDPTW model without modifications could handle these problem classes also.

3.7 Other backhauling problems

Wade and Salhi [45] introduce a problem that generalizes VRPB and MVRPB. In this problem one is not allowed to mix linehaul and backhaul customers on a route freely. A vehicle can only start to serve backhaul customers after a certain percentage of the linehaul load has been delivered. If this percentage is set to 0% then we get the MVRPB and if the percentage is set to 100% then we get the VRPB. Percentages in between 0% and 100% result in a blend between VRPB and MVRPB.

Halskau et al. [17] propose a backhauling problem with so called *lasso* tours. In their problem most customers require both a pickup and a delivery. At the first few customers visited on a route a delivery is performed to free up some room in the vehicle, at the customers in the middle of the route, the delivery and pickup operation is performed simultaneously. The tour is ended by visiting the first couple of customers again, this time in the reverse order to perform the omitted pickups. This creates a tour that looks like a lasso, as the first customers that are visited twice form the spoke of the lasso, while the customers that are visited once form the loop of the lasso.

These two problem variants cannot be solved by the heuristic presented in this paper in its present form. It would only require minor modifications to the heuristic and the underlying model to be able to solve these problems though.

4 Problem transformations

This section describes how each of the problems discussed in Section 3.1–3.6 can be transformed to a Rich PDPTW. The basic transformation is to represent a linehaul customer by a request with a pickup at the depot and a delivery at the linehaul customer. Backhaul customers are represented by a request with a pickup at the backhaul customer and a delivery at the depot. This transformation might seem sufficient to represent the MVRPB but it has the flaw that it allows a vehicle to go back to the depot for re-stocking or offloading and afterwards continue its duty. This is not allowed in a standard MVRPB. The problem is easily solved by assigning precedences to the different tasks: pickups at the depot get precedence 1, deliveries at linehaul customers and pickups at backhaul customers get precedence 2 and deliveries at the depot get precedence 3.

The backhaul after linehaul constraint (A) found in VRPB is also easily modeled using precedences. Instead of giving linehaul and backhaul customers identical precedences, we assign precedence 2 to the linehaul deliveries, precedence 3 to the backhaul pickups and precedence 4 to the deliveries at the depot.

In the VRPB we have to use a specified number of vehicles as stated by constraint (D). Our model only allows us to set an upper bound on the number of vehicles, so we need to model a vehicle equality constraint. This is done by modifying the distance matrix by setting the distance from the start terminal to the end terminal of each vehicle to M, where M is a sufficiently large number. This forces the heuristic towards solutions with at least one request on each route in order to avoid the penalty M.

The VRPB constraint (B) saying that no route can consist of backhauls only, is handled in a similar way. Here we add the penalty M to the cost of each edge from a start terminal to one of the backhaul pickup locations. This drives the heuristic towards solutions where such edges are not used, which means that at least one linehaul customer is served before a backhaul customer.

The simultaneous delivery and pickup constraint in VRPSDP is also modeled using penalties. As before, the delivery to a customer is modeled by a request from the depot to the customer and a pickup at a customer is modeled as a request going from the customer to the depot. In order to ensure that the delivery and pickup occur "simultaneously" we modify the distance matrix. The distance from a delivery visit to the simultaneous pickup visit is set to zero, while the distances from the pickup to all other visits are increased by the penalty term M. This forces the heuristic to visit the simultaneous pickup after a delivery. The situation is illustrated on Figure 3.

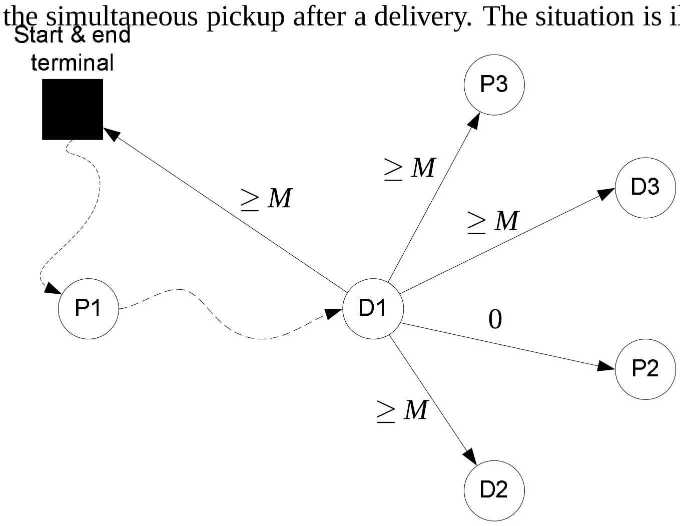


Figure 3: Modeling of simultaneous delivery and pickup. Request 1 is a delivery to a customer, request 2 represents the simultaneous pickup at the same customer and request 3 is another unrelated request. The names "Px" denotes "the pickup of request x" and "Dx" denotes "the delivery of request x". Edge weights are the distances d_{ij}. In order to ensure that D1 is followed by P2 we increase all other distances from D1 with M, while the distance from D1 to P2 is set to zero. In this way, the algorithm will first visit the pickup site of request 1 (the depot) and then travel to the delivery site of request 1 (the customer site). We might perform other visits along the dashed edges. After performing the delivery of request 1, only one edge has cost less than M, hence we go to P2 which is the simultaneous pickup.

The multiple depots in the MDMVRPB are harder to model even though the underlying PDPTW model already supports multiple depots. The problem is that we until now have modeled a linehaul customer by a pickup at the depot and a delivery at the customer, and vice-versa for the backhaul customers. In the multi depot problems we cannot assign a request to a given depot in advance as we do not know where the pickup of a linehaul request or the delivery of a backhaul request should occur. To model this kind of constraint we do the following. For each vehicle in the problem (remember that in the MDMVRPB a fixed number of vehicles is available in each depot) we create a dummy request with pickup and delivery locations at the depot of the vehicle. There is no demand associated with the dummy requests. A dummy request should only be served by the vehicle it is designed for, which is ensured by letting its feasible set of vehicles F_i contain that one vehicle only. We still represent each linehaul customer by one request. All pickups of these linehaul requests take place at a virtual depot. All distances to and from the virtual depot are set to zero. Backhaul customers are represented in the same way — by a pickup at the backhaul customer and a delivery at the virtual depot. The idea is that linehaul requests should travel via the dummy pickup location and backhaul requests should travel via the dummy delivery location. This is ensured using precedences: Linehaul pickups get precedence 1, pickups of the dummy requests get precedence 2, linehaul deliveries and backhaul pickups get precedence 3, deliveries of the dummy requests get precedence 4 and linehaul deliveries get precedence 5. This forces the dummy request to "surround" the linehaul deliveries and backhaul pickups such that the distance to and from the right depot is used. Figure 4 shows an example of a MDMVRPB route with two linehaul customers and one backhaul customer.

A remark should be made about penalty based modeling: If a feasible solution exists that does not violate any of the constraints, the optimal solution will not contain any of the penalty terms. However, since we use heuristics for solving the model, we may end up with a solution which still contains some penalties. This can easily be detected by inspecting the objective value and the heuristic can either be repeated (hoping that a second run will find a better solution) or some manual adjustment of the data may be needed, e.g. by increasing the number of vehicles or by removing some customers which cannot be handled. It should, however, be pointed out that the heuristic has never produced any infeasible solutions during the computational experiments performed in Section 6.

We made heavy use of precedences in the transformations described above. The precedences can also be used to speed up the heuristic when faced with the problem types described in this paper. Consider for example the MVRPB where several pickup and deliveries occur at the depot and all permutations of the pickups at the depot within a route are feasible and equally good as long as the deliveries stay fixed (and similarly for the backhaul deliveries). We can use precedences to create an ordering on the pickups and deliveries at the depot such that only one permutation is valid. We enumerate the request from 1 to n. If request i involves a pickup at the depot, then this pickup gets precedence i, if request i involves a delivery at the depot then this delivery gets precedence $i+n+2$. Pickups and deliveries that corresponds to visits at the customers gets precedence $n+1$. The same idea can be used for the five other problems as well.

5 Solution methods

Recent work on local search methods indicate that larger neighborhoods may be needed to solve some difficult optimization problems as shown by e.g. Ahuja et al. [1]. Due to the size of the neighborhoods, various heuristics are generally used to search the neighborhood in order to keep the time complexity at a reasonable level. This means, that the performance of a local search algorithm is limited by the quality of the heuristic that searches the neighborhood. To work around this bottleneck, Ropke [31] proposed to use several heuristics to search the neighborhood, where the frequency of using each heuristic is based on some empirical evidence from the search. An extended version of this heuristic is used to solve our PDPTW model.

The heuristic is based on *Large Neighborhood Search* (LNS) as proposed by Shaw [35] and it has similarities with the *Ruin and Recreate* (R&R) framework proposed by Schrimpf et al. [33]. Our heuristic repeatedly runs through the following steps:

LNS iteration

1 Choose a removal heuristic R and an insertion heuristic I.
2 Remove a number q of requests from the routes using heuristic R.
3 Insert the free requests into the existing routes using heuristic I.
4 Evaluate the objective function of the new solution.
5 If the objective function is improved, accept the new solution. Otherwise accept the new solution with a probability that depends on the increase of the objective function.

The heuristic differs from the ordinary LNS and R&R methods by incorporating several large-neighborhood heuristics, which are applied with a variable frequency controlled by a learning layer. Each insertion or removal heuristic in the LNS heuristic may have various properties. Some heuristics are used to *intensify* the search while other heuristics mainly play the role of *diversifying* the search. In this way, the learning layer not only distributes CPU-time among the various heuristics involved, but also controls the intensification or diversification of the search based on empirical information. This can be seen as an extension of the tabu search methods described by Hertz et al. [21]. One may also see the LNS algorithm as a variant of *Variable Neighborhood Search* (VNS) described

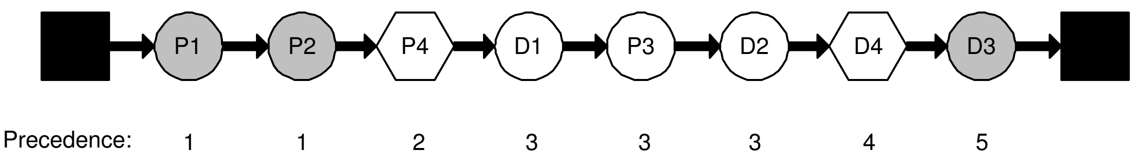


Figure 4: An example of a MDMVRPB route with two linehaul customers and one backhaul customer. The linehaul customers are represented by request 1 and 2 and the backhaul customer is represented by request 3. Request 4 is the dummy request. The start and end terminals are represented by squares, the visits of the normal requests are represented by circles and the visits of the dummy request are represented by hexagons. Pickups and deliveries at the depot are shown in grey and the precedence of the visits is displayed underneath the route. One can observe that the actual MDMVRPB route can be inspected by looking at the white visits; here the hexagons should be viewed as depot visits and the normal deliveries and pickups correspond to the linehaul and backhaul customers respectively.

by Hansen and Mladenovic [18], the main difference being that VNS operates on one type of neighborhood with variable depth, while LNS operates with structurally different neighborhoods.

In the PDPTW heuristic the removal heuristic R removes up to 40% of the requests in each iteration. This enables the heuristic to make significant changes to the current solution in a single iteration. We use six different removal heuristics in our LNS heuristic; each removal heuristic has its own strategy for choosing the requests to remove. The heuristics are:

- *Random removal*: The requests are chosen at random.
- *Shaw removal*: Remove related requests, i.e. requests that are geographically close to each other (Shaw [35]).
- *Worst request removal*: Remove the request whose removal decreases the cost function the most.
- *Cluster removal*: Attempt to partition the nodes into subsets so that the nodes in each subset are somehow “close to each other”. For a more detailed description of this removal heuristics see Section 5.3.
- *History based removal*: This heuristic makes use of historical information when removing requests. Two variants of this heuristic have been considered as will be described in Sections 5.4 and 5.5.

The first three removal heuristics have been used previously [31] while the three last are new.

In order to insert the requests we use the five insertion heuristics proposed by Ropke [31]. The heuristics can be divided into two classes:

- *Basic insertion heuristics*: which are similar to the insertion heuristic of Solomon [37]. In each iteration a request is inserted into the solution such that the cost function is increased the least possible.
- *Regret insertion heuristics*: which are similar to heuristics proposed by Potvin and Rousseau [29] and Tillman and Cain [39]. In each iteration of the standard version of the heuristic a request is inserted so as to maximize the gap in the cost function between inserting the request into its best route and its second best route.

The insertion heuristics are described in more details in [31].

In each step of the PDPTW heuristic one removal and one insertion heuristic are used. Computational experiments have shown that in order to reach high-quality solutions all removal and insertion heuristics are necessary, but their contribution to the solution process may vary during the search.

The *monitoring and learning* layer observes how often a given removal or insertion heuristic contributes to a new, accepted solution, and increases the probability of choosing the given heuristic according to its success. This is done using *roulette wheel selection* where each heuristic has a probability corresponding to its success-rate. In order to ensure that statistical information is collected for all heuristics throughout the search, each heuristic is used not less than a given lower limit.

The LNS algorithm is basically a local search algorithm, and hence it can be combined with most state-of-art local search paradigms. Using the *simulated annealing* paradigm, we evaluate the cost function after each LNS step. If the cost has decreased or is unchanged, the new solution is always accepted. If the cost has increased, the solution is randomly accepted with a probability exponentially decreasing with the increase of the cost.

5.1 Measuring the distance between two requests

In the removal heuristics we need a measure for the distance $d(r_1,r_2)$ between two requests r_1 and r_2. Ropke [31] used the following expression: $d(r_1,r_2) = d_{a_1,a_2} + d_{b_1,b_2}$ where a_1 and a_2 are the pickups of the requests and b_1 and b_2 are the deliveries. This works fine for the pure PDPTW problems but the definition is problematic for backhaul problems. Consider for example two requests corresponding to a linehaul and a backhaul customer located far from the depot. Using the old distance function, the distance between these two requests would be large even though the linehaul and backhaul customer are located close to each another. Instead we use $d(r_1,r_2) = \frac{1}{4}(d_{a_1,a_2} + d_{a_1,b_2} + d_{b_1,a_2} + d_{b_1,b_2})$. If a pickup or a delivery is located at the depot then the distances involving this visit are removed from the formula and the denominator is decremented accordingly.

5.2 Simplified Shaw removal

Shaw [35] defines a removal method that removes related requests. Ropke [31] defines the relatedness between two requests in terms of the distance between the two requests, their capacity demands, temporal information and information about which vehicles can serve the requests. In this paper we take a simpler approach as we define the relatedness between two requests solely by the distance $d(r_1,r_2)$ between the requests.

5.3 Cluster removal

Given a set of points in the plane we can ask to partition the set into $k \geq 2$ disjoint subsets such that the points within each subset are close together with respect to the distance $d(r_1,r_2)$. We say that we partition the points into k clusters.

A heuristic for finding such a partition can be constructed by modifying Kruskal's algorithm [24] for the minimum spanning tree problem. Instead of running Kruskal's algorithm to the end, it can be stopped when k connected components are left. These connected components are our approximation of the desired clusters.

The clustering algorithm is used in a removal heuristic as follows. First a route is selected at random. Then the requests on this route are partitioned into two clusters. One of these clusters is chosen at random and the requests from the chosen cluster are removed. If we need to remove more requests then we pick one of the removed requests and find a request that is close to the chosen request. The new request should come from a route that has not been touched by removals in the current iteration. The route of the new request is partitioned into two clusters and so the process continues until the desired number of requests has been removed. The motivation for the heuristic is to remove large chunks of related requests from a few routes instead of removing a few requests from each route. Figure 5 illustrates when the cluster removal heuristic can be useful.

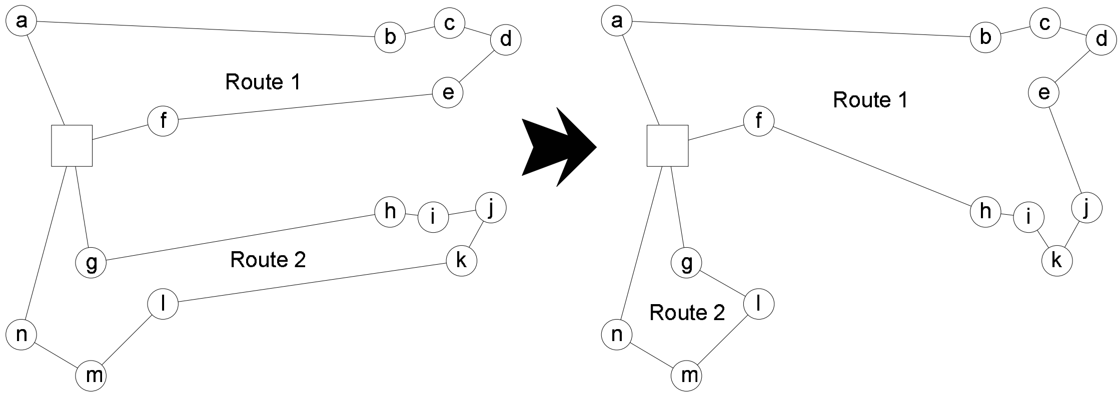


Figure 5: Cluster Removal example: The circles mark the delivery locations, all pickups take place at the depot (marked by the square). In the figure to the left we have a suboptimal solution and we would like to move to the solution shown in the right part of the figure where requests *h*-*k* are placed on the same route as requests *a*-*f*. To reach this solution we need to remove requests *h*, *i*, *j* and *k* at once. If just one of the requests *h*, *i*, *j* or *k* is left on route 2 then the insertion heuristics most likely are going to insert the rest of the requests back into route 2. The removal heuristics presented so far may not be able to remove all of the requests at once, but the cluster removal heuristic does just that. The result of applying the clustering algorithm on route 2 would be the two clusters g,l,m,n and h,i,j,k and the last cluster would be removed with probability 0.5.

5.4 Neighbor graph removal

None of the removal heuristics proposed so far have made any use of historical information when removing requests. The decision about which requests to remove has been made solely by using the information available in the current state.

The *neighbor graph removal heuristic* uses both historical information and the current state to select the requests to remove. The historical information is stored in a complete, directed, weighted graph called the *neighbor graph*.

The graph contains a node for each visit in the problem. The weight of all edges is initially set to plus infinity. The weight of an edge (a,b) stores the cost of the best solution encountered so far in which the visit corresponding to a is performed just before the visit corresponding to b. Each time a new solution is discovered during the search, the edge weights in the graph are updated if necessary.

The graph is used to remove requests that seem to be placed in an unsuitable place. When the removal heuristic is invoked it calculates a score for each request in the current solution. The score is calculated by summing the edge weights in the neighbor graph corresponding to the neighbor configuration in the current solution. The requests with high scores seem to be misplaced and are removed. Every time a request has been removed the scores of the surrounding requests are recalculated. Some randomness is introduced in the removal process in order to avoid removing the same requests over and over again. Specifically the randomness ensures that we sometimes do not remove the requests with the highest score but instead remove some with slightly lower scores.

5.5 Request graph removal

In the request graph removal heuristic we store historical information in a graph called the *request graph*. This graph is complete and undirected and each node in the graph corresponds to a request in the PDPTW problem. The weight of an edge (a,b) denotes the number of times the two requests corresponding to a and b have been served by the same vehicle in the t best unique solutions observed so far in the search. The weights of all edges are initially set to zero, and in all experiments the parameter t was set to 100.

This graph could be used in a similar fashion as the graph described in Section 5.4. That is, we could examine all planned requests r and calculate the score

$$score(r) = \sum_{i \in R(r), i \neq r} w_{ri}$$

where $R(r)$ is the set of requests in the route containing r and w_{ri} is the weight of the edge between r and i in the requests graph. A request with a low score is situated in an unsuitable route according to the request graph and should be removed. Our initial experiments indicated that this was an unpromising approach, probably because it strongly counteracts the diversification mechanisms in the LNS heuristic.

Instead, the graph is used to define the relatedness between two requests, such that two requests are considered to be related if the weight of the corresponding edge in the request graph is high. This relatedness measure is used as in the removal heuristic proposed by Shaw [35], mentioned in Section 5.2.

6 Computational experiments

6.1 Parameter tuning

Even though the proposed heuristic is controlled by quite a few parameters, we have tried to keep the parameter tuning to a minimum in this paper. This is achieved by using the same parameters that were found in the parameter tuning performed by Ropke [31], where applicable. The only parameters that have been tuned are the two parameters that control the simulated annealing: *the cooling rate c* and the *start temperature control parameter w*. After each LNS iteration the temperature T is updated using the recursion $T := cT$. The parameter w controls the start temperature T_0. In order to set the start temperature T_0 we use an estimate of the objective value of a reasonable solution to the problem. This estimate is found by obtaining an initial solution using one of our insertion heuristics and calculating the *modified objective value* z' of this solution. The modified objective value is obtained by setting the coefficient β to zero, such that unplanned requests do not make the estimate of the objective value unreasonably high. Now the start temperature is set such that a solution that is $1+w$ times larger than z' is accepted with probability 0.5 when the current solution has objective z'. We have tested the algorithm on 11 problems chosen from 5 of the 6 problem categories. The configuration $w = 0.05$ and $c = 0.9998$ proved to be the best among the 30 configurations tested. The same parameters were used for all problem types considered in the following sections.

6.2 Test strategy

The LNS heuristic is tested on 9 data sets proposed in the literature. The test serves two major purposes. The first purpose is to compare three configurations of the LNS heuristic against each other. The three configurations are:

- A configuration similar to the one used by Ropke [31]. This configuration benefits from the learning layer but is limited to the 3 "old" removal heuristics: The *simplified Shaw removal*, the *worst removal* and the *random request removal*. This configuration is denoted *standard* in the following.
- A configuration that uses all 6 removal heuristics but has disabled the learning layer. This implies that all removal and insertion heuristics are equally likely to be selected during the search. This configuration is denoted *6R - no learning* in the following (the "6R" indicates that 6 removal heuristics are in use).
- The last configuration is similar to the second, but in the third configuration the learning layer is activated again. The configuration is denoted *6R - normal learning*.

These three configurations allow us to see if the new removal heuristics improve the quality of the heuristic and enable us to judge the effectiveness of the learning layer.

The second major purpose of the test is to compare the solution quality obtained by the unified heuristic to the results obtained by more specialized heuristics proposed for the various problem types. We want to know whether a general heuristic can be competitive with specialized heuristics.

The stopping criterion employed is to stop when the heuristic has performed 25000 remove-insert iterations. Each configuration of the heuristic is applied 10 times to each problem instance. The reported computation times are, however, for a single run of the algorithm.

All problems considered in the following are geometric problems where distances and travel times are defined by the Euclidean distance, hence the triangle inequality is satisfied for both parameters. When it has been necessary to calculate distances from a set of coordinates we have used double precision calculations unless otherwise stated. For many of the problem classes we only present a summary of the experiments performed. We refer the reader to the appendix for the full tables for these problems. All experiments were performed on a Linux based PC, equipped with 256 MB RAM and a 1.5 GHz Pentium IV processor. The heuristic was implemented in C++.

6.3 The Vehicle Routing Problem with Backhauls (VRPB)

The first problem type we study is the symmetric VRPB. This problem along with the VRPBTW is probably the most studied of the backhaul problems. Two data sets are proposed in the literature, the first was proposed by Goetschalckx and Jacobs-Blecha [16] and contains 62 instances with between 20 and 150 customers. The second data set was proposed by Toth and Vigo [40] and contains 33 instances with between 21 and 100 customers. We denote the two data sets the *Goetschalckx* and the *Toth-Vigo* data sets respectively.

Comparing results on the Goetschalckx data set are a little problematic as at least 3 different rounding conventions have been used for calculating the distances between the customers in the data sets. We report our results obtained using 2 of the 3 rounding conventions and refer to the appendix for a discussion about the third rounding convention and the results obtained using it.

Currently the two best heuristics for the VRPB are probably the heuristic proposed by Toth and Vigo [40] and the heuristic by Osman and Wassan [28]. The heuristic by Toth and Vigo finds good solutions in a short time while the heuristic proposed by Osman and Wassan spends more time but on the overall finds better solutions. We compare our heuristic with the results found by Osman and Wassan as the running time of our algorithm is comparable to that of Osman and Wassan's heuristic. In order to calculate the distance between two customers, Osman and Wassan used floating point arithmetic, hence we do the same (using double precision) in the tests reported in Table 1.

The tests show that the configurations using all 6 removal heuristics are better than the one using only three removal heuristics. This test also shows that the configuration that does not include the learning layer overall is slightly better than the configuration including the learning layer, which is a bit surprising. All configurations of the LNS heuristics do better than Osman and Wassan's heuristic when looking at how many best known solutions the heuristics have found. It should be noted that the best solution found by Osman and Wassan's heuristic was found in 8 experiments, while we used 10 experiments for each LNS configuration. If one looks at the sum of the best solution costs identified by the heuristics, it is observed that the LNS heuristics overall only marginally improve the solutions found by Osman and Wassan's heuristic; for all LNS heuristics the improvement is within 0.1%. All together the LNS heuristics improved the solution of 26 of the 62 problem instances. Finally we see that the average solution costs found by the LNS heuristics are quite good as they on average are less than 0.5% from the best known solution costs.

Generally it is hard to compare the running time of our heuristic to that of the heuristics proposed in the literature, as the computational experiments have been performed on different computers. According to the Linpack benchmarks reports [11], our computer has a TPP rating (*Toward Peak Performance*) of 1311 MFlops while Osman and Wassan's Computer has a TPP rating of 25 MFlops, implying that our computer is around 53 times faster. The average time for solving one problem was between 69 and 73 seconds for the LNS heuristics. Osman and Wassan tested two versions of their heuristic, the fastest version using around 2800 seconds to solve one problem and the slower version using 4000 seconds. This corresponds to 52 and 75 seconds on our computer, which is very comparable to the time used by our algorithm. Hence our general heuristic is on par with Osman and Wassan's specialized heuristic both with respect to solution quality and solution times.

The second way to calculate the distances is to round them to one decimal, and store them as an integers using a fixed point representation. The final result is rounded to an integer. This type of rounding is used in the exact methods developed by Toth and Vigo [41] and Mingozzi et al. [26]. 34 of the 62 instances have been solved to optimality and a good solution is provided for 13 more problems without proving optimality. Table 2 summarizes the results obtained by applying the heuristic to these 47 problems (problem *A1-K4*) using the same rounding conventions as the exact methods. These results also show that the configurations that use the new removal heuristics are better than the one that only uses the 3 old removal heuristics. This time the configurations with and without the learning layer are virtually equally good. All configurations find 28 optimal solutions out of the 34 optimal solutions reported by Toth and Vigo [41] and Mingozzi et al. [26]. Eight new best solutions were found in the tests.

The *Toth-Vigo* data set have been approached by the exact methods of Toth and Vigo [41] and Mingozzi et al. [26] and by the heuristics of Crispim and Brandao [7], Osman and Wassan [28] and Toth and Vigo [40]. Table 3 reports the results found by the LNS heuristic compared with the best known results from the literature. We see that the configuration with learning enabled provides the best solutions on the average; furthermore it is the only one which identifies all known optimal solutions. The configuration without learning overall finds slightly better solutions compared to the learning version when summing the best solution from the ten experiments. The LNS heuristics improve the best known solutions to 5 of the problems.

A class of asymmetric problem instances was proposed by Toth and Vigo [41], but we have not included this data set in our test even though our PDPTW model would be able to handle the asymmetric problems.

6.4 The Mixed Vehicle Routing Problem with Backhauls (MVRPB)

Two data sets have been proposed for the MVRPB. The first set is based on a relaxed version of the *Goetschalckx* problems, and it has been studied by Halse [19] and Wade and Salhi [43], [44]. The other data set, which was proposed by Nagy and Salhi [27], is constructed by transforming 14 well-known CVRP instances into MVRPB instances. Three MVRPB instances are constructed from each CVRP instance, having 10%, 25% and 50% of the customers transformed to backhaul customers. Heuristics are applied to the last data set by Dethloff [9] and Nagy and Salhi [27], [32]. We decided to test our heuristic on MVRPB by using the last data set.

The chosen data set contains 42 problems with 50 to 199 customers. Table 4 compares the solutions obtained by the LNS heuristics to the solutions obtained by Nagy and Salhi. Unfortunately it is not possible to include the results obtained by Dethloff [9] in the table as Dethloff only tested his algorithm on a subset of the problems. The heuristic named NS1 in the table is a construction algorithm and the heuristic named NS2 is a construction heuristic followed by an improvement algorithm. Both are much faster than the LNS heuristics. The comparison shows that

	Best known		Standard				6R - no learning				6R - normal learning			
	n	cost	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
A1	25	**229885.65**	229885.65	**229885.65**	0.00	7	229885.65	**229885.65**	0.00	7	229885.65	**229885.65**	0.00	7
A2	25	**180119.21**	180119.21	**180119.21**	0.00	8	180119.21	**180119.21**	0.00	8	180119.21	**180119.21**	0.00	8
A3	25	**163405.38**	163405.38	**163405.38**	0.00	9	163405.38	**163405.38**	0.00	10	163405.38	**163405.38**	0.00	9
A4	25	**155796.41**	155796.41	**155796.41**	0.00	10	155796.41	**155796.41**	0.00	10	155796.41	**155796.41**	0.00	11
B1	30	**239080.15**	239080.16	239080.16	0.00	9	239080.16	239080.16	0.00	9	239080.16	239080.16	0.00	9
B2	30	**198047.77**	198047.77	**198047.77**	0.00	10	198047.77	**198047.77**	0.00	10	198047.77	**198047.77**	0.00	10
B3	30	**169372.29**	169372.29	**169372.29**	0.00	13	169372.29	**169372.29**	0.00	14	169372.29	**169372.29**	0.00	14
C1	40	**250556.77**	250846.82	**250556.77**	0.12	14	250560.15	**250556.77**	0.00	14	250556.77	**250556.77**	0.00	13
C2	40	**215020.23**	215020.23	**215020.23**	0.00	16	215020.23	**215020.23**	0.00	16	215020.23	**215020.23**	0.00	16
C3	40	**199345.96**	199345.96	**199345.96**	0.00	18	199345.96	**199345.96**	0.00	20	199345.96	**199345.96**	0.00	18
C4	40	**195366.63**	195366.63	**195366.63**	0.00	19	195366.63	**195366.63**	0.00	19	195366.63	**195366.63**	0.00	19
D1	38	**322530.13**	322530.13	**322530.13**	0.00	12	322530.13	**322530.13**	0.00	12	322530.13	**322530.13**	0.00	12
D2	38	**316708.86**	316708.86	**316708.86**	0.00	11	316708.86	**316708.86**	0.00	12	316708.86	**316708.86**	0.00	12
D3	38	**239478.63**	239478.63	**239478.63**	0.00	13	239478.63	**239478.63**	0.00	13	239478.63	**239478.63**	0.00	13
D4	38	**205831.94**	205831.94	**205831.94**	0.00	16	205831.94	**205831.94**	0.00	16	205831.94	**205831.94**	0.00	15
E1	45	**238879.58**	238879.58	**238879.58**	0.00	18	238879.58	**238879.58**	0.00	18	238879.58	**238879.58**	0.00	18
E2	45	**212263.11**	212463.34	**212263.11**	0.09	23	212263.11	**212263.11**	0.00	23	212458.75	**212263.11**	0.09	22
E3	45	**206659.17**	206710.33	**206659.17**	0.02	26	206697.72	**206659.17**	0.02	27	206761.96	**206659.17**	0.05	26
F1	60	**264299.6**	268346.03	267060.43	1.53	31	268430.58	267060.43	1.56	30	268306.24	267060.43	1.52	29
F2	60	265653.47	265214.16	**265214.16**	0.00	29	265214.16	**265214.16**	0.00	29	265214.16	**265214.16**	0.00	28
F3	60	**241120.77**	241969.77	241969.77	0.35	37	241969.77	241969.77	0.35	36	241969.77	241969.77	0.35	35
F4	60	**233861.85**	235175.20	235175.20	0.56	43	235528.13	235175.20	0.71	44	235449.66	235175.20	0.68	42
G1	57	**306305.4**	306388.11	**306305.40**	0.03	23	306322.98	**306305.40**	0.01	23	306354.90	**306305.40**	0.02	22
G2	57	**245440.99**	245529.35	**245440.99**	0.04	29	245440.99	**245440.99**	0.00	28	245440.99	**245440.99**	0.00	27
G3	57	**229507.48**	229507.48	**229507.48**	0.00	33	230737.17	**229507.48**	0.54	32	230583.46	**229507.48**	0.47	30
G4	57	235251.47	232913.81	**232521.25**	0.17	32	233006.36	**232521.25**	0.21	32	233263.98	**232521.25**	0.32	31
G5	57	**221730.35**	221826.32	**221730.35**	0.04	35	222435.96	**221730.35**	0.32	36	222442.67	**221730.35**	0.32	35
G6	57	**213457.45**	213541.70	**213457.45**	0.04	40	214090.55	**213457.45**	0.30	42	213457.45	**213457.45**	0.00	39
H1	68	**268933.06**	269342.45	**268933.06**	0.15	41	269467.78	**268933.06**	0.20	42	269317.64	**268933.06**	0.14	39
H2	68	**253365.5**	253423.34	**253365.50**	0.02	49	253462.09	**253365.50**	0.04	49	254194.18	**253365.50**	0.33	47
H3	68	**247449.04**	247532.87	**247449.04**	0.03	56	247508.59	**247449.04**	0.02	55	247449.04	**247449.04**	0.00	53
H4	68	**250220.77**	250317.37	**250220.77**	0.04	52	250269.07	**250220.77**	0.02	53	250269.07	**250220.77**	0.02	52
H5	68	**246121.31**	246532.25	**246121.31**	0.17	58	246767.73	**246121.31**	0.26	58	246217.90	**246121.31**	0.04	55
H6	68	**249135.32**	249294.67	**249135.32**	0.06	55	249231.92	**249135.32**	0.04	57	249206.96	**249135.32**	0.03	55
I1	90	351606.91	350958.02	350258.81	0.20	55	350852.85	**350245.28**	0.17	54	350897.94	350247.61	0.19	52
I2	90	309955.04	312489.95	**309943.84**	0.82	66	311016.93	**309943.84**	0.35	65	310434.77	**309943.84**	0.16	63
I3	90	**294507.38**	295236.14	**294507.38**	0.25	86	294858.13	**294507.38**	0.12	83	294821.76	**294507.38**	0.11	81
I4	90	295999.65	296820.65	**295988.45**	0.28	79	296159.12	**295988.45**	0.06	77	296401.46	**295988.45**	0.14	76
I5	90	302524.33	302707.04	**301236.01**	0.49	76	301909.59	**301236.01**	0.22	75	301980.98	**301236.01**	0.25	74
J1	95	335593.42	336680.78	**335006.68**	0.50	60	336522.31	**335006.68**	0.45	58	336789.92	335479.75	0.53	56
J2	95	310800.53	312206.97	**310417.21**	0.58	71	312458.56	**310417.21**	0.66	67	311763.08	**310417.21**	0.43	65
J3	95	**279219.21**	281807.92	**279219.21**	0.93	94	279423.74	**279219.21**	0.07	87	279729.03	**279219.21**	0.18	84
J4	95	296773.38	298412.68	297232.88	0.63	77	297781.22	**296533.16**	0.42	74	297344.74	297086.58	0.27	72
K1	113	395546.4	397774.56	394846.98	0.86	86	395993.78	**394375.63**	0.41	83	397076.46	395006.60	0.68	81
K2	113	363214.24	365791.18	362656.70	1.01	100	362998.61	**362130.00**	0.24	97	363253.47	**362130.00**	0.31	96
K3	113	366222.05	367806.64	**365694.08**	0.58	99	366218.02	**365694.08**	0.14	97	366388.14	**365694.08**	0.19	95
K4	113	349038.84	351441.74	**348949.39**	0.71	113	349266.17	**348949.39**	0.09	111	349241.78	**348949.39**	0.08	108
L1	150	426017.86	428037.41	**426013.41**	0.48	162	427658.80	**426013.41**	0.39	153	427641.03	426281.89	0.38	149
L2	150	402245.17	402073.43	401466.27	0.21	192	401587.25	**401228.80**	0.09	181	401492.36	401247.70	0.07	176
L3	150	403886.22	404784.84	**402677.72**	0.52	187	403029.19	**402677.72**	0.09	176	402860.67	**402677.72**	0.05	174
L4	150	384844.01	387660.68	**384636.33**	0.79	220	385207.32	**384636.33**	0.15	207	385073.14	**384636.33**	0.11	205
L5	150	388061.69	390091.24	**387564.55**	0.65	210	388677.62	**387564.55**	0.29	211	389778.12	**387564.55**	0.57	200
M1	125	400860.79	402962.88	401006.99	1.02	108	401540.39	**398913.70**	0.66	104	401666.48	**398913.70**	0.69	102
M2	125	398908.71	400924.09	399001.11	0.53	108	401724.68	399336.27	0.73	102	401347.29	**398827.67**	0.63	100
M3	125	377352.81	379362.69	377411.62	0.85	122	378502.30	377212.23	0.62	115	378031.96	**376159.13**	0.50	114
M4	125	348624.42	349984.33	348624.42	0.45	147	348663.06	**348417.94**	0.07	140	348905.97	348532.69	0.14	137
N1	150	**408926.4**	414655.53	409210.18	1.40	162	414044.03	410789.32	1.25	156	414915.65	410419.05	1.46	155
N2	150	**409280.16**	413434.54	410595.02	1.02	164	413124.59	409385.19	0.94	155	415985.72	411131.25	1.64	153
N3	150	396167.85	402418.80	398841.27	2.05	181	399363.23	**394337.86**	1.27	177	400984.40	396827.00	1.69	170
N4	150	397753.86	401362.13	397363.45	1.67	178	402131.56	398965.12	1.86	172	400553.31	**394788.36**	1.46	170
N5	150	376431.84	380168.38	375895.96	1.79	222	377447.83	**373476.30**	1.06	214	378201.49	375201.45	1.27	210
N6	150	377665.19	381099.86	377368.09	1.96	216	376612.61	**373758.65**	0.76	211	376966.15	373789.70	0.86	209
Tot.		18058230	18124900	18055590		4536	18093048	18042916		4405	18098312	18044860		4299
Avg.					0.43	73			0.29	71			0.31	69
BTPB			20				24				23			
#B	36		43				53				46			

Table 1: *Goetschalckx* problems. The table compares the results obtained by the three configurations of the LNS heuristics with the best results obtained by Osman and Wassan's heuristic [28]. The two first columns show the problem name and the number of customers in the problem. The third column displays the best solution found by Osman and Wassan's heuristic. The rest of the columns are divided into three sections, one for each configuration. These should be interpreted as follows: *avg. sol.* - the average of the solution costs obtained in the 10 experiments, *best sol.* - the cost of the best solution found in the 10 experiments, *avg. gap (%)* - the gap between average and best known solution cost, *avg. time (s)* - the average time needed to perform one experiment (in seconds). The best solution for each problem instance is marked with bold. The row *Tot.* at the bottom of the table gives the sum of the given column and the row *Avg.* gives the average of the column. The row *BTPB* reports the number of problem instances where a particular configuration found solutions that were better than the previous best known solution, the row *#B* contains the number of times the heuristic found the best known solution to a problem.

	Avg. gap (%)	#B	Avg. time (s)	Opt.	BTPB
Standard	0.28	35	39	28	8
6R - no learning	0.18	38	40	28	8
6R - normal learning	0.17	36	40	28	8

Table 2: Summary of testing the 47 first *Goetschalckx* problems using distances rounded to one decimal. Each row in the table corresponds to one of the three LNS configurations. The columns *Avg. gap (%)* and *Avg. Time (s)* should be interpreted like the corresponding entries in the *Avg.* row in Table 1. The rest of the columns are: *#B* - the number of problems where the best known solution was reached, *Opt.* the number of optimal solutions found (out of 34 known optimal solutions), *BTPB* - the number of problems for which the heuristic improved the solutions found by the branch and bound methods. The improved solutions correspond to problems were the branch and bound algorithms did not reach optimality because they were stopped before optimality was proved.

	Best known				Standard				6R - no learning				6R - normal learning			
	n	cost	opt	reference	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
EIL22.50A	21	**371**	X	TV + EHP	371	**371**	0.00	8	371	**371**	0.00	8	371	**371**	0.00	8
EIL22.66A	21	**366**	X	TV + EHP	366	**366**	0.00	7	366	**366**	0.00	8	366	**366**	0.00	7
EIL22.80A	21	**375**	X	TV + EHP	375	**375**	0.00	7	375	**375**	0.00	8	375	**375**	0.00	8
EIL23.50A	22	**682**	X	TV + EHP	709	**682**	3.94	13	682	**682**	0.00	12	682	**682**	0.00	12
EIL23.66A	22	**649**	X	TV + EHP	654	**649**	0.77	12	649	**649**	0.00	13	649	**649**	0.00	13
EIL23.80A	22	**623**	X	TV + EHP	625	**623**	0.26	11	623	**623**	0.00	12	623	**623**	0.00	12
EIL30.50A	29	**501**	X	TV + EHP	501	**501**	0.00	17	501	**501**	0.00	19	501	**501**	0.00	18
EIL30.66A	29	**537**	X	TV + EHP	537	**537**	0.00	13	537	**537**	0.00	14	537	**537**	0.00	14
EIL30.80A	29	**514**	X	TV + EHP	514	**514**	0.00	13	514	**514**	0.00	14	514	**514**	0.00	14
EIL33.50A	32	**738**	X	TV + EHP	738	**738**	0.00	17	738	**738**	0.00	20	738	**738**	0.00	20
EIL33.66A	32	**750**	X	TV + EHP	750	**750**	0.00	15	750	**750**	0.00	17	750	**750**	0.00	16
EIL33.80A	32	**736**	X	TV + EHP	737	**736**	0.18	15	736	**736**	0.05	15	736	**736**	0.05	15
EIL51.50A	50	**559**	X	TV + EHP	561	**559**	0.41	35	559	**559**	0.00	39	559	**559**	0.00	36
EIL51.66A	50	**548**	X	TV + EHP	553	**548**	0.91	30	550	**548**	0.35	31	549	**548**	0.11	30
EIL51.80A	50	**565**	X	TV + EHP	569	**565**	0.65	28	571	**565**	1.12	29	570	**565**	0.80	28
EILA76.50A	75	**739**	X	TV + EHP	740	**739**	0.16	49	739	**739**	0.00	50	739	**739**	0.00	48
EILA76.66A	75	**768**	X	TV + EHP	774	**768**	0.77	44	774	769	0.73	44	772	**768**	0.51	42
EILA76.80A	75	**781**		TV + EHP	794	783	1.63	41	794	783	1.72	40	791	783	1.22	39
EILB76.50A	75	**801**	X	TV + EHP	804	**801**	0.31	42	802	**801**	0.12	42	803	**801**	0.25	40
EILB76.66A	75	**873**	X	TV + EHP	876	**873**	0.38	38	875	**873**	0.22	38	873	**873**	0.01	37
EILB76.80A	75	**919**	X	TV + EHP	927	**919**	0.90	36	924	**919**	0.58	38	922	**919**	0.37	37
EILC76.50A	75	**713**	X	TV + EHP	715	**713**	0.21	60	713	**713**	0.04	61	713	**713**	0.00	59
EILC76.66A	75	**734**	X	EHP	740	735	0.75	51	739	**734**	0.69	51	736	**734**	0.23	50
EILC76.80A	75	**733**		TV + EHP	738	734	0.71	48	741	736	1.09	48	738	737	0.70	47
EILD76.50A	75	**690**	X	TV + EHP	702	**690**	1.77	71	696	**690**	0.81	75	691	**690**	0.20	71
EILD76.66A	75	**715**		TV + EHP	717	**715**	0.22	59	716	**715**	0.20	60	715	**715**	0.00	57
EILD76.80A	75	**694**		EHP	699	**694**	0.72	53	699	695	0.76	55	696	**694**	0.26	53
EILA101.50A	100	842		OSMAN	845	837	1.72	138	840	**831**	1.05	137	836	**831**	0.55	129
EILA101.66A	100	**846**	X	TV + EHP	852	**846**	0.67	99	848	**846**	0.21	100	846	**846**	0.05	99
EILA101.80A	100	875		OSMAN	872	862	1.77	91	869	**857**	1.41	87	866	861	1.03	86
EILB101.50A	100	933		EHP	930	**925**	0.54	82	928	**925**	0.31	79	929	**925**	0.38	77
EILB101.66A	100	998		OSMAN	1007	994	1.79	69	1010	**989**	2.13	66	1001	991	1.24	66
EILB101.80A	100	1021		OSMAN	1022	1018	1.43	63	1021	1010	1.26	61	1015	**1008**	0.65	61
Tot.		23189			23314	23160		1373	23251	23140		1394	23201	23142		1349
Avg.							0.71	42			0.45	42			0.26	41
BTPB					5				5				5			
#B	28				26				28				29			

Table 3: *Toth-Vigo* data set. The column *opt* indicates if optimality is proven for the particular instance and the column *reference* points to the algorithm that found the solution in the *best known* column. *TV* refers to the exact method by Toth and Vigo [41], *EHP* refers to the exact algorithm by Mingozzi et al. [26] and *OSMAN* refers to the heuristic by Osman and Wassan [28].

	NS1	NS2	Standard	6R - no learning	6R - normal learning
10%	1011	995	956 (3.9%)	**955 (4.0%)**	956 (3.9%)
25%	1034	998	923 (7.5%)	923 (7.5%)	**922 (7.6%)**
50%	1045	991	**881 (11.1%)**	**881 (11.1%)**	**881 (11.1%)**

Table 4: Summary of the 42 Nagy-Salhi MVRPB problem instances. This table compares the solutions obtained by the LNS heuristic to those obtained by Nagy and Salhi [27], [32]. Each row reports the average solution over 14 MVRPB instances with a particular percentage of backhaul customers (10%, 25% or 50%). The columns NS1 and NS2 contain the best results reported by Nagy and Salhi in [32] and [27] respectively. The last three columns show the results obtained by the LNS heuristic. The numbers in parenthesis show how much better the LNS solutions are compared to the solutions reported by Nagy and Salhi.

	Standard				6R - no learning				6R - normal learning			
	Avg. gap (%)	#B	BTPB	Avg. time (s)	Avg. gap (%)	#B	BTPB	Avg. time (s)	Avg. gap (%)	#B	BTPB	Avg. time (s)
10%	0.51	10	13	129	0.43	11	13	133	0.37	11	13	133
25%	0.49	11	14	135	0.38	9	14	142	0.30	11	14	143
50%	0.71	7	13	164	0.45	10	14	178	0.41	12	13	178

Table 5: This table provides a comparison of the 3 LNS configurations when applied to the 42 Nagy-Salhi MVRPB instances. Each row summarizes 14 instances with the same percentage of backhaul customers. The meaning of the headings is as in Table 2.

great improvements can be achieved by using a more advanced heuristics such as the LNS heuristic proposed here, as we get results that are more than 10% better than those obtained by the simpler heuristics. We succeeded in improving the best known solution for 41 out of the 42 problems. On the last problem we matched the solution reported by Nagy and Salhi. Notice that the average solution cost decreases when more customers are turned into backhaul customers in the solutions provided by the LNS heuristic. This is expected as a greater percentage of backhaul customers leads to greater flexibility in the planning as long as the percentage of backhaul customers is not greater than 50%. It is worth noting that Nagy and Salhi's results do not show this behavior.

Table 5 compares the three LNS configurations. The results show that the configurations with six removal heuristics overall are better than the one with three removal heuristics when one compares the gaps. The results also show that the configuration with the learning layer enabled is better than the one without the learning layer. One can also notice that the computation time increases as more customers are turned into backhaul customers. This behavior can most likely be explained by the fact that routes in general contain many customers when the percentage of backhauls customers is around 50%. Long routes imply that more time is spent in the insertion heuristics.

6.5 The Multiple Depot Mixed Vehicle Routing Problem with Backhauls (MDMVRPB)

Only one data set has been proposed for the MDMVRPB. This data set was proposed by Nagy and Salhi [32] and is constructed from Gillett and Johnson's 11 multi depot vehicle routing problems. Each of the 11 problems are turned into three MDMVRPB problems by creating problems with 10%, 25% and 50% backhaul customers; thus the MDMVRPB data set contains 33 problems with between 50 and 249 customers. The only heuristics that have been applied to the problems so far are those by Nagy and Salhi which also were used for the MVRPB discussed in Section 6.4.

In Table 6 we compare the results obtained by the LNS heuristic with those obtained by the best heuristics of Nagy and Salhi [27], [32]. It has been necessary to reconstruct the problems from Gillett and Johnson's original problems following the description in [32], as the original problems no longer were available from the authors. We believe that the problems have been constructed properly. The reconstructed problems have been made available on the web [46] for future comparisons. Again, we observe that the LNS heuristic offers huge improvements over

	NS1	NS2	Standard	6R - no learning	6R - normal learning
10%	2008	1996	1798 (9.9%)	**1795 (10.1%)**	1799 (9.9%)
25%	2050	2007	1671 (16.7%)	1663 (17.1%)	**1662 (17.2%)**
50%	2088	1993	1512 (24.1%)	1510 (24.2%)	**1509 (24.3%)**

Table 6: Summary of results obtained on the 33 Nagy-Salhi MDMVRPB instances. The columns NS1 and NS2 contain the best results reported by Nagy and Salhi in [32] and [27] respectively.

	Standard				6R - no learning				6R - normal learning			
	Avg. gap (%)	#B	BTPB	Avg. time (s)	Avg. gap (%)	#B	BTPB	Avg. time (s)	Avg. gap (%)	#B	BTPB	Avg. time (s)
10%	0.93	7	11	204	0.63	10	11	217	0.61	6	11	216
25%	0.97	5	11	219	0.65	6	11	237	0.66	8	11	237
50%	0.88	8	11	258	0.71	6	11	288	0.66	7	11	288

Table 7: Nagy-Salhi MDMVRPB instances. Comparison of the performance of the three LNS configurations.

the simpler heuristics. This time the solution costs are decreased by up to 24% and the best known solutions to all problems were improved. As before we note that the heuristics proposed by Nagy and Salhi are faster than the LNS heuristic.

Table 7 compares the three LNS configurations with each other. The most interesting observation is that the multi depot problems seem to be the hardest problems considered so far, as the average solutions are farther from the best known solutions than before, but the results must anyway be considered as very promising.

6.6 The Vehicle Routing Problem with Backhauls and Time Windows (VRPBTW)

The VRPBTW is another well-studied backhauling problem. The primary objective considered in the heuristics described in the literature is to minimize the number of vehicles used and the secondary objective is to minimize the traveled distance. These objectives are also used in our experiments. The vehicle minimization is done by solving the problem for a decreasing number of vehicles, as proposed by Ropke [31]. Gelinas et al. [13] proposed a data set containing 15 problems with 100 customers and Thangiah et al. [38] introduced a data set containing 24 large problems.

Our heuristics are tested on both data sets. The results obtained on Gelinas' data set are presented in Table 8. Five papers have reported results on this data set: Duhamel et al. [12], Hasama et al. [20], Reimann et al. [30], Thangiah et al. [38] and Zhong and Cole [48]. It should be noted that apparently there is no standard for how distances should be represented internally in the heuristic, which makes comparisons a bit problematic. We have chosen to represent the distances using doubles like Reimann et al. [30], as is standard in the literature about VRPTW heuristics. The tables reveal that we are able to improve 10 out of the 15 solutions and reduce the number of vehicles needed for 5 of the problems. Again the configurations using all removal heuristics turns out to be the best.

The only heuristic that has been applied to the large VRPBTW problems is the heuristic by Thangiah et al. [38]. Table 9 compares the results obtained by this algorithm to the results obtained by the LNS heuristic. We see that the LNS heuristic is able to decrease the necessary number of vehicles by a large amount and at the same time also decrease the traveled distance. The best known solutions to all 24 problems were improved by the LNS heuristic. Table 10 gives further information about the performance of the LNS heuristic, including the running time. The time increases with the problem size, but its growth is not alarming. Once again the configurations using 6 removal heuristics found the best solutions.

	Best known				Standard				6R - no learning				6R - normal learning			
	% BH	m	cost	ref	avg. #veh.	best sol.	best #veh.	avg. time (s)	avg. #veh.	best sol.	best #veh.	avg. time (s)	avg. #veh.	best sol.	best #veh.	avg. time (s)
BHR101A	10%	22	1831.68	RDH	22.0	**1818.86**	**22**	98	22.0	**1818.86**	**22**	107	22.0	**1818.86**	**22**	109
BHR101B	30%	23	1999.16	RDH	23.0	1959.86	23	94	23.0	**1959.56**	**23**	101	23.0	**1959.56**	**23**	103
BHR101C	50%	**24**	**1909.84**	HKK	24.0	1939.10	24	93	24.0	1939.10	24	100	24.0	1939.10	24	101
BHR102A	10%	19	1677.62	RDH	19.0	**1653.19**	**19**	110	19.0	**1653.19**	**19**	118	19.0	**1653.19**	**19**	121
BHR102B	30%	**21**	**1764.3**	TPS	22.0	1750.70	22	103	22.0	1750.70	22	111	22.0	1750.70	22	114
BHR102C	50%	**21**	**1745.7**	TPS	22.0	1775.76	22	103	22.0	1775.76	22	111	22.0	1775.76	22	113
BHR103A	10%	**15**	**1371.6**	TPS	15.0	1387.57	15	117	15.0	1387.57	15	123	15.0	1387.57	15	128
BHR103B	30%	16	1395.88	RDH	15.0	**1390.33**	**15**	108	15.0	**1390.33**	**15**	112	15.0	**1390.33**	**15**	115
BHR103C	50%	**16**	**1486.56**	ZC	17.0	1457.31	17	106	17.0	1456.48	17	113	17.0	1456.48	17	115
BHR104A	10%	11	1205.78	RDH	11.0	1084.22	11	127	11.0	**1084.17**	**11**	130	11.0	**1084.17**	**11**	132
BHR104B	30%	12	1128.3	RDH	11.0	1163.24	11	119	11.0	**1154.84**	**11**	121	11.0	**1154.84**	**11**	122
BHR104C	50%	12	1208.46	RDH	11.0	1191.41	11	117	11.0	**1191.38**	**11**	119	11.0	**1191.38**	**11**	119
BHR105A	10%	16	1544.81	RDH	15.5	1564.88	15	104	15.3	**1561.28**	**15**	110	15.4	**1561.28**	**15**	109
BHR105B	30%	16	1592.23	RDH	16.0	**1583.30**	**16**	97	16.0	**1583.30**	**16**	102	16.0	**1583.30**	**16**	102
BHR105C	50%	17	1633.01	RDH	16.6	1711.36	16	96	16.6	1710.75	16	100	16.5	**1710.19**	**16**	100
Tot.		261	23495		260.2	23432	259	1593	260.0	23418	259	1679	259.9	23417	259	1703
Avg.								106				112				114
BTPB					10				10				10			
#B	5				4				9				10			

Table 8: The table shows the results obtained on the VRPBTW instances proposed by Gelinas et al. [13]. The first column shows the name of the problem, the next columns are: *%BH* - ratio of backhaul customers, *m* - number of vehicles in best known solution, *cost* - distance traveled in best known solution, *ref* - HKK = Hasama et al. [20], RDH = Reimann et al. [30], TPS = Thangiah et al. [38] and ZC = Zhong and Cole [48], the result found by Zhong and Cole was listed in their technical report [47]. The rest of the columns report the solutions found by the LNS heuristics: *avg. #veh.* - average number of vehicles *best #veh.* - lowest number of vehicles found. The other columns should be interpreted as in Table 1. The original data files do not specify the latest return time to the depot and the maximum capacity of the vehicle. In our experiments these parameters have been set to the values they have in the original Solomon problems from which the Gelinas problems were created.

	TPS		Standard		6R - no learning		6R - normal learning	
	#veh.	cost	#veh.	cost	#veh.	cost	#veh.	cost
250	517	57509	449	54256	**444**	**54711**	445	54499
500	799	94144	677	83498	676	82946	**675**	**82796**

Table 9: Large VRPBTW instances. This table compares the 3 LNS configurations to the heuristics by Thangiah et al. (TPS). The data set contains 12 problems containing 250 customers and 12 containing 500 customers. The best solutions found by the heuristics have been accumulated and the table shows the total number of vehicles needed and the total traveled distance for all instances of a particular size. The vehicle capacity was set to 200 for all problems and no latest arrival time was specified for the depot.

	Standard				6R - no learning				6R - normal learning			
Customers	Avg. #veh.	#B	BTPB	Avg. time (s)	Avg. #veh.	#B	BTPB	Avg. time (s)	Avg. #veh.	#B	BTPB	Avg. time (s)
250	37.5	1	12	489	37.3	6	12	492	37.4	5	12	504
500	57.1	0	12	1562	56.8	4	12	1651	56.7	8	12	1570

Table 10: Comparison of the three LNS configurations when faced with the large VRPBTW instances proposed by Thangiah. The *Avg. #veh* column displays the average of the average number of vehicles needed to serve all customers.

	LB	KB		ZC		Standard		6R - no learning		6R - normal learning	
	#veh.	#veh.	cost	#veh.	cost	#veh.	cost	#veh.	cost	#veh.	cost
MR2	4	4	1168.53	4	1016.66	4	904.55	**4**	**902.73**	4	903.00
MC2	4	4	1094.94	4.625	903.56	**4**	**731.38**	4	732.38	4	732.13
MRC2	4	4.5	1496.91	4.125	1330.31	**4.125**	**1125.00**	4.125	1129.25	4.125	1127.63

Table 11: Kontoravdis MVRPBTW problems. The table compares the results reported by Kontoravdis and Bard [23] (KB) and Zhong and Cole [48] (ZC) with the results obtained using the LNS heuristics. The primary objective in these problems is to minimize the number of vehicles needed to serve the customers. The data set is divided into three classes according to the geographical distribution of the customers in the problems: randomly distributed customers (MR2), clustered customers (MC2), and a mix between the two first categories (MRC2). The MRC2 and MC2 classes both contain 8 problems while the MR2 class contains 11 problems. Each row in the table summarizes the performance on each class. The column *LB #veh.* shows the lower bound on the number of vehicles as given by Kontoravdis and Bard.

	Standard				6R - no learning				6R - normal learning			
	Avg. gap (%)	#B	BTPB	Avg. time (s)	Avg. gap (%)	#B	BTPB	Avg. time (s)	Avg. gap (%)	#B	BTPB	Avg. time (s)
MR2	1.34	4	11	362	0.63	8	11	375	0.63	8	11	368
MC2	0.62	6	8	162	0.60	5	8	165	0.65	5	8	163
MRC2	2.83	5	8	183	1.99	1	8	183	1.76	4	8	180

Table 12: The table compares the three LNS configurations when applied to Kontoravdis' MVRPBTW problems. In all test runs the heuristics reached the same number of vehicles when applied to the same problem. This allows us to report the *avg. gap*, which doesn't make sense if the heuristics use a different number of vehicles to solve the same problem.

6.7 The Mixed Vehicle Routing Problem with Backhauls and Time Windows (MVRPBTW)

Two datasets have been proposed for the MVRPBTW. Hasama et al. [20] use Gelinas' data set by relaxing the linehaul-before-backhaul constraint while Kontoravdis and Bard [23] construct 27 new problems from Solomon's VRPTW problems. We test our heuristics using Kontoravdis and Bard's data set which also has been attempted by Zhong and Cole [48]. The LNS heuristic is compared to the previous heuristics in Table 11. Again the LNS heuristic is able to find solutions of better quality compared to the older heuristics. It is interesting to note that the LNS heuristic reaches the lower bound on the number of vehicles needed to solve the problems on all but one instance. The LNS heuristics improved all the previously best known solutions to the problem instances.

Table 12 provides the usual comparison of the three LNS configurations. It should be observed that the MRC2 problems turn out to be hard to solve, as indicated by the rather large gaps. This is not surprising as the MRC2 problems were constructed from Solomon's RC2 VRPTW problems, which are known to be hard to solve. One cannot expect that adding the extra complexity of backhaul customers should make the problems easier to solve.

6.8 The Vehicle Routing Problem with Simultaneous Deliveries and Pickups (VRPSDP)

Allthough the VRPSDP is not the problem in the backhauling family that has received the most attention, there exist nevertheless quite a few data sets for the problem. The first data set was proposed by Min [25] and contained only one problem, which originated from a real life application. Halse [19] proposed a set containing 16 problems constructed from CVRP problems and Dethloff [10] proposed 40 new problems containing 50 customers each. Nagy and Salhi [32] constructed two classes of VRPSDP problems and two classes of multi depot VRPSDP problems. Finally Angelelli and Mansini [2] presented a class of VRPSDP problems with time windows.

As mentioned earlier we are not going to test our heuristic on the multi depot and time window variants of the VRPSDP. The problems we choose for our tests are Min's problem, Dethloff's problems and the first class of Nagy and Salhi's VRPSDP problems (the one denoted with an X in [32]). The results are summarized in Tables 13 and 14. Again it must be stressed that the heuristics by Dethloff and Nagy and Salhi are simple construction heuristics

	Dethloff	NS1	NS2	Standard	6R - no learning	6R - normal learning
Dethloff	824	-	-	747 (9.3%)	746 (9.5%)	**745 (9.6%)**
NS-X	1006	1096	991	927 (6.5%)	925 (6.7%)	**919 (7.3%)**

Table 13: Summary of the results obtained on the VRPSDP instances. The table should be interpreted like Table 4. The row denoted *Dethloff* summarizes the results obtained on Dethloff's 40 instances [10] and the single instance provided by Min [25]. Each of Dethloff's instances contains 50 customers. The row marked *NS-X* summarizes Nagy and Salhi's 14 VRPSDP instances of class *X* [32]. These problems contain between 50 and 200 customers. Results for these problems are reported by Dethloff [10] and Nagy and Salhi [32], [27]. The columns *Dethloff*, *NS1* and *NS2* summarize the best results reported in [10], [32] and [27] respectively.

	Standard				6R - no learning				6R - normal learning			
	Avg. gap (%)	#B	BTPB	Avg. time (s)	Avg. gap (%)	#B	BTPB	Avg. time (s)	Avg. gap (%)	#B	BTPB	Avg. time (s)
Dethloff	1.07	24	40	128	0.96	23	40	129	0.58	36	40	155
NS-X	2.81	6	11	685	2.73	7	13	686	2.00	7	12	772

Table 14: The table compares the 3 LNS configurations when applied to VRPSDP instances.

that are substantially faster than the LNS heuristics.

The LNS heuristics find the optimal solution to Min's problem (the optimal solution was found by Halse [19]) and are able to improve all of the best known solutions to Dethloff's problems which were found using Dethloff's construction heuristic. The *6R - normal learning* configuration is able to improve the best known solutions by more than 9%. Having said that, it should be noticed that the LNS heuristics are fairly slow when faced with this type of problems, because each order is represented by 2 requests and introduces significant overhead in the algorithm. This also suggests that this problem type would benefit greatly from a specialized version of the LNS heuristic where the overhead can be avoided. The LNS heuristic also experiences difficulties when faced with the larger problems from Nagy and Salhi's data set. Here the avg. gap increases to 2% for the best configuration, but the heuristic nevertheless improves 13 of the 14 best known solutions. The configuration with learning enabled and using all 6 removal heuristics clearly is the most robust configuration when faced with these hard problems.

6.9 Computational experiments conclusion

In Section 6.2 we raised a number of questions that the computational experiments should clarify. The first question was whether it is possible to design a unified heuristic for a large class of vehicle routing problems with backhauls that is able to provide solutions comparable to those obtained by specialized heuristics. We believe that the experiments conducted in this paper show that this indeed is possible. This is an interesting achievement, as it to a large extent allows practitioners to focus on a single heuristic and apply this to the problems they are faced with instead of "reinventing the wheel" each time a new problem type needs to be solved.

The second question asked to give an evaluation of the effect of the three new removal heuristics and the consequence of disabling the learning layer. Table 15 provides an overview of the experiments performed. The *Avg.* row displays the overall gaps between average solutions and best known solutions. This gap is an indication of the robustness of the heuristic. The *Sum* row contains the number of problems for which the particular LNS configuration found the best known solution. The table clearly shows the impact of adding the three new removal heuristics, as we see a great improvement in the quality of the heuristic from configuration 1 to configuration 3. The table also shows that disabling the learning layer decreases the overall quality of the results as expected. Although comparable results can be obtained without the learning layer for specific problem types, the learning layer apparently helps the algorithm to adapt to all the various problem types.

	#prob	Standard		6R - no learning		6R - normal learning	
		Avg. gap (%)	#B	Avg. gap (%)	#B	Avg. gap (%)	#B
Goetschalckx 1	62	0.43	43	0.29	53	0.31	46
Goetschalckx 2	47	0.28	35	0.17	38	0.17	36
Toth-Vigo	33	0.71	26	0.45	28	0.26	29
MVRPB 50%	14	0.71	7	0.45	10	0.41	12
MVRPB 25%	14	0.49	11	0.38	9	0.3	11
MVRPB 10%	14	0.51	10	0.43	11	0.37	11
MDMVRPB 50%	11	0.88	8	0.71	6	0.66	7
MDMVRPB 25%	11	0.97	5	0.65	6	0.66	8
MDMVRPB 10%	11	0.93	7	0.63	10	0.61	6
VRPSDP 1	41	1.07	24	0.96	23	0.58	36
VRPSDP 2	14	2.81	5	2.73	7	2.00	7
MVRPBTW C	8	0.63	6	0.6	5	0.65	5
MVRPBTW R	11	1.34	4	0.63	8	0.63	8
MVRPBTW RC	8	2.83	5	1.99	1	1.76	3
VRPBTW 1	15		4		9		10
VRPBTW 2	24		1		10		13
Avg.		0.81		0.62		0.50	
Sum	338		201		234		248

Table 15: Summary of experiments. This table shows a summary of the tests performed in this paper. Each row in the table corresponds to a problem class. Most of the titles in the first row should be fairly self explanatory: *Goetschalckx 1* - Goetschalckx VRPB without rounding distances, *Goetschalckx 2* - Goetschalckx VRPB where distances have been rounded to one decimal. *VRPSDP 1* - Dethloff VRPSDP , *VRPSDP 2* - Nagy-Salhi VRPSDP, *VRPBTW 1* - Gelinas VRPBTW *VRPBTW 2* - Thangiah VRPBTW. The column *#prob* displays the number of problems in each class. The *Avg.* row shows the averages of the *Avg. gap(%)* column. The numbers in the avg. row were calculated by summing the products of the numbers in the *#prob* column with the numbers in the *gap* column and dividing the sum by the total number of problems. This was done to take into account that some data sets contains more problems than others. The missing entries in the VRPBTW rows have been left out because the primary objective of these problems is to minimize the number of vehicles and not all test runs resulted in the same number of vehicles. Reporting the gap for these runs could make the heuristic that could not reach the minimum number of vehicles look too good.

7 Conclusion

This paper is the first to present a unified heuristic for a large class of vehicle routing problems with backhauls. For this purpose we have introduced a Rich VRPTW model which extends the ordinary VRP model with time windows, pickup and delivery pairs, as well as precedence constraints. The model is very expressive, and it allows us to model all of the most common VRPB models within the framework, as well as other routing problems from the literature. The unified model has the additional benefit that it allows us to combine pickup and delivery request with a more clean VRPB or VRPSPD, as well as scheduling mixed transportation problems for a general fleet of vehicles.

For several of the VRPB problem types presented in this paper, we report the first applications of a metaheuristic to the problem. The results are very promising as we found a new best solution to 67% of the problems tested. Even faster and better performing heuristics could be constructed by specializing the proposed heuristic to just one of the problem types. We have chosen not to do this to maintain the generality of the solution approach.

The present experiments indicate that the combination of several neighborhoods makes it easier for the local search heuristic to explore the solution space, and hence to find solutions of high quality. This conforms to similar observations for simpler neighborhoods.

The monitoring and learning layer to control the choice of neighborhoods can be seen as a layer which maintains a proper balance between intensification and diversification. Several other approaches have been working with this balance, see e.g. Reactive Tabu Search [4]. In the proposed framework we do not explicitly care about which heuristics intensify or diversify the search. The layer steadily maintains a proper balance of the heuristics so that new, improved solutions are found. The computational results show that the learning layer overall is able to increase the robustness of the heuristic but also indicate that further refinements may be possible as the configuration without the learning layer occasionally outperformed the configuration that included the learning layer.

An interesting topic for further research would be to apply the framework proposed in this paper to combinatorial optimization problems outside the vehicle routing domain.

8 Acknowledgements

The authors wish to thank Jan Dethloff, George Kontoravdis, Marc Reimann, Sam R. Thangiah and Daniele Vigo for kindly providing us with the data sets used in this paper and for answering questions regarding the data sets. Furthermore we wish to thank Jakob Birkedal Nielsen for proposing the Cluster Removal heuristic and Gabor Nagy for sending us his working paper.

9 Appendix

This section contains additional information about the experiments performed in section 6. Tables 16 to 31 list the individual solutions found to the many problem instances considered in this paper.

An important comment should be made about Table 17. The results in this table were obtained by rounding distances to the nearest integer when doing distance calculations. This gives results that look like the results reported in Table III in Osman and Wassan [28] and Table 1 in Toth and Vigo [40] but both author pairs state that results in these tables were found using a different rounding procedure. We have not been able to reproduce the results in the two mentioned tables from Toth and Vigo and Osman and Wassan papers using the rounding procedures described in the papers. Consequently, the objective values listed in the column *Best known* in table 17 should only be seen as a rough guideline of the obtainable solution quality, and the table should not be used to make a direct comparison between the LNS heuristic and the heuristics by Toth and Vigo and Osman and Wassan.

	Best known				Std. Removals				6R - no learning				6R - normal learning			
	n	cost	opt.	reference	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
A1	25	**229886**	X	TV + EHP	229886	**229886**	0.00	7	229886	**229886**	0.00	7	229886	**229886**	0.00	7
A2	25	**180119**	X	TV + EHP	180119	**180119**	0.00	7	180119	**180119**	0.00	8	180119	**180119**	0.00	8
A3	25	**163405**	X	TV + EHP	163405	**163405**	0.00	9	163405	**163405**	0.00	10	163405	**163405**	0.00	9
A4	25	**155796**	X	TV + EHP	155796	**155796**	0.00	10	155796	**155796**	0.00	11	155796	**155796**	0.00	10
B1	30	**239080**	X	TV + EHP	239080	**239080**	0.00	8	239080	**239080**	0.00	9	239080	**239080**	0.00	9
B2	30	**198048**	X	TV + EHP	198048	**198048**	0.00	10	198048	**198048**	0.00	10	198048	**198048**	0.00	10
B3	30	**169372**	X	TV + EHP	169372	**169372**	0.00	12	169372	**169372**	0.00	14	169372	**169372**	0.00	14
C1	40	**249449**	X	TV + EHP	250899	250557	0.58	13	251037	250557	0.64	14	250557	250557	0.44	13
C2	40	**215020**	X	TV + EHP	215020	**215020**	0.00	15	215020	**215020**	0.00	16	215020	**215020**	0.00	16
C3	40	**199346**	X	TV + EHP	199346	**199346**	0.00	17	199346	**199346**	0.00	18	199346	**199346**	0.00	18
C4	40	**195366**	X	TV + EHP	195366	**195366**	0.00	18	195366	**195366**	0.00	19	195366	**195366**	0.00	19
D1	38	**322530**	X	TV + EHP	322530	**322530**	0.00	11	322530	**322530**	0.00	12	322530	**322530**	0.00	12
D2	38	**316709**	X	TV + EHP	316709	**316709**	0.00	11	316709	**316709**	0.00	13	316709	**316709**	0.00	12
D3	38	**239479**	X	EHP	239479	**239479**	0.00	12	239479	**239479**	0.00	13	239479	**239479**	0.00	12
D4	38	**205832**	X	EHP	205832	**205832**	0.00	14	205832	**205832**	0.00	15	205832	**205832**	0.00	15
E1	45	**238880**	X	TV + EHP	238880	**238880**	0.00	16	238880	**238880**	0.00	18	238880	**238880**	0.00	18
E2	45	**212263**	X	TV + EHP	212547	**212263**	0.13	21	212263	**212263**	0.00	23	212505	**212263**	0.11	24
E3	45	**206659**	X	TV + EHP	206698	**206659**	0.02	24	206698	**206659**	0.02	27	206711	**206659**	0.03	26
F1	60	**263173**	X	TV + EHP	268334	267060	1.96	28	268463	267060	2.01	29	268321	267060	1.96	29
F2	60	**265213**	X	TV + EHP	265213	**265213**	0.00	27	265213	**265213**	0.00	28	265213	**265213**	0.00	28
F3	60	**241120**	X	TV + EHP	241969	241969	0.35	33	241969	241969	0.35	35	241969	241969	0.35	35
F4	60	**233861**	X	TV + EHP	236547	235175	1.15	40	235258	235175	0.60	42	235449	235175	0.68	42
G1	57	**306305**	X	EHP	306450	306306	0.05	21	306306	306306	0.00	22	306306	306306	0.00	22
G2	57	**245441**	X	EHP	245441	**245441**	0.00	27	245441	**245441**	0.00	27	245441	**245441**	0.00	27
G3	57	**229507**	X	TV	229536	**229507**	0.01	30	230430	**229507**	0.40	30	230003	**229507**	0.22	30
G4	57	**232521**	-	EHP	232784	**232521**	0.11	29	233767	**232521**	0.54	31	233649	**232521**	0.48	31
G5	57	**221730**	X	TV	221805	**221730**	0.03	33	221771	**221730**	0.02	35	221730	**221730**	0.00	36
G6	57	**213457**	X	TV	213562	**213457**	0.05	38	213457	**213457**	0.00	41	214084	**213457**	0.29	39
H1	68	**268933**	X	TV	269701	**268933**	0.29	38	269276	**268933**	0.13	40	269371	**268933**	0.16	40
H2	68	**253365**	X	TV + EHP	253414	**253365**	0.02	45	253437	**253365**	0.03	48	253365	**253365**	0.00	47
H3	68	**247449**	X	TV + EHP	247684	**247449**	0.10	51	247474	**247449**	0.01	53	247475	**247449**	0.01	54
H4	68	**250221**	X	TV + EHP	250244	**250221**	0.01	49	250221	**250221**	0.00	52	250295	**250221**	0.03	51
H5	68	**246121**	X	TV + EHP	247300	**246121**	0.48	56	246170	**246121**	0.02	57	246140	**246121**	0.01	55
H6	68	**249135**	X	TV + EHP	249397	**249135**	0.11	53	249246	**249135**	0.04	54	249246	**249135**	0.04	55
I1	90	353021	-	EHP	351106	350437	0.25	52	350951	**350246**	0.20	52	351069	350801	0.24	52
I2	90	**309943**	X	EHP	311714	309944	0.57	63	310738	309944	0.26	63	310846	309944	0.29	63
I3	90	294833	-	EHP	296221	**294507**	0.58	80	294728	**294507**	0.07	84	294950	**294507**	0.15	81
I4	90	**295988**	-	EHP	296889	**295988**	0.30	75	296172	**295988**	0.06	75	296374	**295988**	0.13	76
I5	90	**301226**	-	EHP	302666	301236	0.48	71	301619	301236	0.13	74	302066	301236	0.28	73
J1	95	**335006**	-	EHP	336598	335007	0.48	57	336475	335480	0.44	58	336347	335007	0.40	57
J2	95	315644	-	EHP	311853	**310417**	0.46	65	311440	**310417**	0.33	66	310964	**310417**	0.18	67
J3	95	282447	-	EHP	282335	280401	1.12	83	279801	**279219**	0.21	86	279468	**279219**	0.09	84
J4	95	300548	-	EHP	298004	296773	0.50	72	297529	**296533**	0.34	74	297249	**296533**	0.24	72
K1	113	394637	-	EHP	398657	**394376**	1.09	82	397183	**394376**	0.71	83	395965	394517	0.40	82
K2	113	362360	-	EHP	364447	**362130**	0.64	96	363103	**362130**	0.27	98	363258	**362130**	0.31	95
K3	113	**365693**	-	EHP	367725	365694	0.56	95	366549	365694	0.23	97	366698	365694	0.27	96
K4	113	358308	-	EHP	352064	**348950**	0.89	108	349775	**348950**	0.24	109	349483	**348950**	0.15	107
Tot.		12174445			12188672	12157811		1832	12172829	12156670		1902	12171434	12156893		1881
Avg.							0.28	39			0.18	40			0.17	40
< PB						8				8				8		
#B		39				35				38				36		

Table 16: *Goetschalckx* data set. The results have been produced by using distances rounded to one decimal and rounding the final result to an integer. This rounding scheme allows us to compare the LNS heuristics to the exact methods by Toth and Vigo [41] and Mingozzi et al. [26], we only report results on the instances that either Toth and Vigo or Mingozzi et al. attempted to solve. The table should be read like Table 3, notice that the row *<PB* should be interpretted like the *BTPB* row in Table 3.

	Best known			Std. Removals				6R - no learning				6R - normal learning			
	n	cost	reference	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
A1	25	**229884**	TV	229884	**229884**	0.00	7	229884	**229884**	0.00	8	229884	**229884**	0.00	8
A2	25	**180117**	TV	180117	**180117**	0.00	8	180117	**180117**	0.00	9	180117	**180117**	0.00	8
A3	25	**163403**	TV	163403	**163403**	0.00	9	163403	**163403**	0.00	10	163403	**163403**	0.00	10
A4	25	**155795**	TV	155795	**155795**	0.00	10	155795	**155795**	0.00	11	155795	**155795**	0.00	11
B1	30	**239077**	TV	239077	**239077**	0.00	9	239077	**239077**	0.00	10	239077	**239077**	0.00	9
B2	30	**198045**	TV	198045	**198045**	0.00	10	198045	**198045**	0.00	11	198045	**198045**	0.00	11
B3	30	**169368**	TV	169368	**169368**	0.00	13	169368	**169368**	0.00	15	169368	**169368**	0.00	15
C1	40	**250557**	TV	250557	**250557**	0.00	13	250557	**250557**	0.00	14	250557	**250557**	0.00	14
C2	40	**215019**	TV	215019	**215019**	0.00	15	215019	**215019**	0.00	17	215019	**215019**	0.00	17
C3	40	**199344**	TV	199344	**199344**	0.00	18	199344	**199344**	0.00	20	199344	**199344**	0.00	21
C4	40	**195365**	TV	195365	**195365**	0.00	18	195365	**195365**	0.00	20	195365	**195365**	0.00	20
D1	38	**322533**	TV	322533	**322533**	0.00	11	322533	**322533**	0.00	13	322533	**322533**	0.00	13
D2	38	**316711**	TV	316711	**316711**	0.00	11	316711	**316711**	0.00	13	316711	**316711**	0.00	12
D3	38	**239482**	TV	239482	**239482**	0.00	12	239482	**239482**	0.00	14	239482	**239482**	0.00	13
D4	38	**205834**	TV	205834	**205834**	0.00	15	205834	**205834**	0.00	16	205834	**205834**	0.00	16
E1	45	**238880**	TV	238880	**238880**	0.00	17	238880	**238880**	0.00	19	238880	**238880**	0.00	19
E2	45	**212262**	TV	212262	**212262**	0.00	23	212262	**212262**	0.00	24	212262	**212262**	0.00	24
E3	45	**206658**	TV	206734	**206658**	0.04	25	206709	**206658**	0.02	28	206722	**206658**	0.03	28
F1	60	**263175**	TV	268435	267061	2.00	30	267941	267061	1.81	31	268242	267061	1.93	31
F2	60	**265214**	TV	265230	**265214**	0.01	29	265214	**265214**	0.00	30	265214	**265214**	0.00	30
F3	60	**241121**	OW	242014	241970	0.37	36	241970	241970	0.35	38	241970	241970	0.35	37
F4	60	**233861**	TV	235912	235178	0.88	42	235261	235178	0.60	45	235204	235178	0.57	44
G1	57	**306304**	OW	306455	**306304**	0.05	22	306336	**306304**	0.01	23	306304	**306304**	0.00	24
G2	57	**245441**	TV	245533	**245441**	0.04	28	245441	**245441**	0.00	29	245441	**245441**	0.00	29
G3	57	**229506**	OW	229963	**229506**	0.20	32	230421	**229506**	0.40	33	230414	**229506**	0.40	32
G4	57	232646	TV	233142	**232519**	0.27	31	233951	**232519**	0.62	33	233705	**232519**	0.51	33
G5	57	**221731**	OW	221823	**221731**	0.04	36	221858	**221731**	0.06	38	221800	**221731**	0.03	39
G6	57	**213457**	TV	213605	**213457**	0.07	41	213457	**213457**	0.00	43	213516	**213457**	0.03	43
H1	68	**268933**	OW	269630	**268933**	0.26	40	269460	**268933**	0.20	43	269226	**268933**	0.11	42
H2	68	**253366**	TV	253513	**253366**	0.06	48	253463	**253366**	0.04	50	253414	**253366**	0.02	50
H3	68	**247449**	TV	247803	**247449**	0.14	54	247594	**247449**	0.06	57	247472	**247449**	0.01	57
H4	68	**250221**	TV	250449	**250221**	0.09	51	250269	**250221**	0.02	56	250269	**250221**	0.02	55
H5	68	**246121**	TV	246367	**246121**	0.10	57	246265	**246121**	0.06	61	246339	**246121**	0.09	60
H6	68	**249136**	TV	249280	**249136**	0.06	54	249284	**249136**	0.06	60	249187	**249136**	0.02	59
I1	90	351609	OW	351136	350437	0.25	54	350902	**350248**	0.19	55	350992	**350248**	0.21	55
I2	90	309957	OW	312017	**309946**	0.67	66	311039	**309946**	0.35	66	310739	**309946**	0.26	66
I3	90	**294509**	OW	295043	**294509**	0.18	86	294788	**294509**	0.09	88	294773	**294509**	0.09	88
I4	90	**295988**	TV	296414	**295988**	0.14	79	296370	**295988**	0.13	80	296254	**295988**	0.09	84
I5	90	302525	OW	302482	**301238**	0.41	75	301916	**301238**	0.23	78	302225	**301238**	0.33	80
J1	95	335590	OW	336867	**335004**	0.56	60	336418	335478	0.42	61	336243	**335004**	0.37	60
J2	95	310798	OW	312248	**310417**	0.59	70	311378	**310417**	0.31	71	311662	**310417**	0.40	70
J3	95	**279220**	OW	281860	279307	0.95	90	279830	**279220**	0.22	91	279889	**279220**	0.24	92
J4	95	296774	OW	297926	296861	0.47	77	297487	**296533**	0.32	79	297436	**296533**	0.30	78
K1	113	395544	OW	397824	394511	0.88	87	396806	394458	0.62	88	395328	**394369**	0.24	87
K2	113	363213	OW	365244	362358	0.86	102	363938	**362128**	0.50	103	363350	**362128**	0.34	102
K3	113	366222	OW	368228	**365693**	0.69	100	366593	**365693**	0.25	102	366420	**365693**	0.20	101
K4	113	349037	OW	351283	**348947**	0.67	115	349713	**348947**	0.22	116	349191	**348947**	0.07	114
L1	150	426021	OW	427532	**426014**	0.36	162	427786	426283	0.42	162	428031	426178	0.47	160
L2	150	402246	OW	402643	**401231**	0.35	196	401917	401426	0.17	192	401720	**401231**	0.12	188
L3	150	403886	OW	404400	**402681**	0.43	189	402829	**402681**	0.04	189	402681	**402681**	0.00	187
L4	150	384843	OW	388152	**384635**	0.91	221	384962	**384635**	0.08	219	385656	**384635**	0.27	218
L5	150	388060	OW	392003	**387563**	1.15	216	388986	**387563**	0.37	215	388398	**387563**	0.22	214
M1	125	400858	OW	403542	**400085**	0.86	109	402393	400660	0.58	109	402158	401076	0.52	108
M2	125	398902	OW	400668	398712	0.81	109	400842	399263	0.85	109	400262	**397448**	0.71	106
M3	125	377352	OW	379724	377139	0.70	123	378814	377399	0.46	121	378548	**377093**	0.39	121
M4	125	348624	OW	349623	348604	0.31	149	349083	**348530**	0.16	149	349331	**348530**	0.23	146
N1	150	**408921**	OW	416448	409897	1.84	166	414350	410046	1.33	165	413239	409506	1.06	163
N2	150	**409275**	OW	415947	410232	1.63	167	415411	410232	1.50	163	415049	410616	1.41	163
N3	150	**396162**	OW	401857	396870	1.44	185	401359	396825	1.31	182	400977	397546	1.22	180
N4	150	397748	OW	402257	398293	1.89	180	399558	**394785**	1.21	180	401012	398667	1.58	181
N5	150	376426	OW	380571	377081	1.90	229	375915	**373471**	0.65	227	378486	374553	1.34	222
N6	150	377660	OW	379159	375646	1.45	221	378089	**373752**	1.16	226	376755	375348	0.80	225
Tot.		18053986		18130660	18051840		4555	18096042	18044295		4629	18092915	18048852		4598
Avg.						0.45	73			0.30	75			0.28	74
< PB					20				20				20		
#B		39			45				49				51		

Table 17: *Goetschalckx* data set. The results have been produced by using distances rounded to integers. The results in the *best known* columns were found by the heuristics proposed by Toth and Vigo (TV) [40] and Osman and Wassan (OW) [28]. Notice that the TV and OW heuristics might have used a different rounding procedure, and consequently this table cannot be used to compare the LNS heuristics to the two earlier heuristics (see the text in the appendix). The table is provided for future reference only.

	Best known		Std. Removals				6R - no learning				6R - normal learning			
	cost	Reference	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
CMT01T	541	NS	520	**520**	0.00	32	520	**520**	0.00	34	520	**520**	0.00	34
CMT02T	839	NS	790	**783**	0.95	52	792	784	1.13	56	788	**783**	0.63	57
CMT03T	903	NS	805	801	0.83	104	804	801	0.72	110	803	**798**	0.65	109
CMT04T	1111	NS	1004	**998**	0.63	203	1004	**998**	0.61	213	1005	1000	0.73	212
CMT05T	1423	NS	1239	1231	0.97	323	1239	1232	0.95	334	1234	**1227**	0.57	333
CMT06T	571	NS	555	**555**	0.00	29	555	**555**	0.00	31	555	**555**	0.00	31
CMT07T	-	-	909	**903**	0.69	48	907	**903**	0.38	52	904	**903**	0.16	52
CMT08T	911	NS	869	**866**	0.43	91	868	**866**	0.33	94	866	**866**	0.10	95
CMT09T	**1164**	NS	1172	1166	0.75	173	1170	**1164**	0.56	179	1172	**1164**	0.67	178
CMT10T	1418	NS	1410	1398	1.09	285	1408	**1395**	0.93	291	1410	1402	1.05	291
CMT11T	1075	NS	1002	**999**	0.29	158	1001	**999**	0.24	163	1003	1000	0.39	164
CMT12T	827	NS	789	**788**	0.14	92	788	**788**	0.00	96	788	**788**	0.00	96
CMT13T	1600	NS	1550	**1544**	0.35	124	1548	**1544**	0.23	126	1547	**1544**	0.21	127
CMT14T	866	NS	827	**827**	0.06	84	827	**827**	0.00	86	827	**827**	0.00	86
Tot.	13249		13442	13378		1801	13430	13375		1864	13422	13376		1864
Avg.					0.51	129			0.43	133			0.37	133
< PB				13				13				13		
#B		1		10				11				11		

Table 18: Nagy and Salhi MVRPB problems with 10% backhaul customers. The entries in the *Best known* columns are the best result reported by Nagy and Salhi (NS) [32] and Dethloff (D) [9]. It should be noted that Dethloff's heuristic only have been applied to half of the problems. No solution were given for problem 7 (this explains the dash in the table).

	Best known		Std. Removals				6R - no learning				6R - normal learning			
	cost	Reference	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
CMT01Q	557	NS	490	**490**	0.02	35	490	**490**	0.00	40	490	**490**	0.00	41
CMT02Q	860	NS	737	**732**	0.62	57	736	733	0.54	64	737	733	0.64	65
CMT03Q	918	NS	752	**747**	0.68	119	751	**747**	0.58	126	749	**747**	0.23	128
CMT04Q	1164	NS	922	**916**	0.60	228	921	918	0.58	244	922	918	0.59	244
CMT05Q	1477	NS	1133	1124	1.35	358	1127	**1118**	0.83	382	1124	1119	0.52	381
CMT06Q	594	NS	555	**555**	0.00	28	555	**555**	0.00	30	555	**555**	0.00	30
CMT07Q	-	-	905	**901**	0.44	48	903	**901**	0.26	52	902	**901**	0.17	53
CMT08Q	918	NS	868	**866**	0.25	90	867	**866**	0.23	93	866	**866**	0.10	93
CMT09Q	1178	NS	1170	**1162**	0.69	167	1170	1164	0.69	170	1169	**1162**	0.62	171
CMT10Q	1477	NS	1404	1394	1.06	280	1405	1398	1.11	285	1402	**1389**	0.91	288
CMT11Q	1075	NS	941	**939**	0.22	183	941	**939**	0.23	195	941	**939**	0.12	196
CMT12Q	843	NS	731	**729**	0.28	100	731	**729**	0.21	107	730	**729**	0.17	108
CMT13Q	1613	NS	1554	1545	0.67	117	1546	1544	0.14	120	1546	**1543**	0.14	120
CMT14Q	873	NS	822	**822**	0.00	84	822	**822**	0.00	85	822	**822**	0.00	85
Tot.	13547		12983	12922		1896	12966	12924		1993	12954	12914		2003
Avg.					0.49	135			0.38	142			0.30	143
< PB				14				14				14		
#B		0		11				9				11		

Table 19: Nagy Salhi MVRPB problems with 25% backhaul customers. See Table 18 for a decription.

	Best known		Std. Removals				6R - no learning				6R - normal learning			
	cost	Reference	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
CMT01H	536	D	468	466	0.63	44	465	**465**	0.08	50	466	**465**	0.19	51
CMT02H	801	D	666	**663**	0.47	69	664	**663**	0.21	76	664	**663**	0.21	78
CMT03H	850	D	705	**701**	0.62	165	702	**701**	0.14	183	702	**701**	0.11	186
CMT04H	1099	D	842	835	1.57	306	840	**829**	1.27	346	840	**829**	1.24	345
CMT05H	1329	D	996	986	1.36	461	994	986	1.12	514	991	**983**	0.78	514
CMT06H	595	NS	555	**555**	0.00	29	555	**555**	0.00	31	555	**555**	0.00	31
CMT07H	-	-	904	901	0.40	49	902	901	0.23	52	903	**900**	0.29	54
CMT08H	915	NS	866	**866**	0.08	92	867	**866**	0.14	94	868	**866**	0.26	95
CMT09H	1164	NS	1169	1164	0.72	172	1171	**1161**	0.86	176	1169	1166	0.69	177
CMT10H	1509	NS	1406	**1389**	1.24	290	1406	1396	1.23	295	1401	1393	0.92	296
CMT11H	961	D	829	**818**	1.37	271	820	**818**	0.26	315	818	**818**	0.04	303
CMT12H	765	D	636	630	1.00	135	633	**629**	0.65	146	635	**629**	0.86	150
CMT13H	1546	NS	1552	1544	0.54	120	1545	1544	0.13	123	1546	**1543**	0.18	125
CMT14H	866	NS	822	**822**	0.00	87	822	**822**	0.00	89	822	**822**	0.00	89
Tot.	12936		12416	12338		2291	12387	12335		2490	12379	12333		2493
Avg.					0.71	164			0.45	178			0.41	178
< PB				13				14				13		
#B		0		7				10				12		

Table 20: Nagy Salhi MVRPB problems with 50% backhaul customers. See Table 18 for a decription.

	Best known		Std. Removals				6R - no learning				6R - normal learning			
	cost	Reference	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
GJ01T	614	NS	570	**569**	0.12	30	569	**569**	0.00	34	569	**569**	0.00	35
GJ02T	497	NS	464	**464**	0.04	34	464	**464**	0.04	37	464	**464**	0.00	38
GJ03T	662	NS	627	**624**	0.34	60	626	**624**	0.29	65	626	625	0.19	64
GJ04T	1055	NS	976	972	1.43	80	969	**962**	0.75	85	971	**962**	0.92	86
GJ05T	794	NS	739	735	0.85	114	738	**733**	0.62	118	738	**733**	0.61	119
GJ06T	914	NS	859	**851**	0.90	85	853	**851**	0.21	90	852	**851**	0.16	91
GJ07T	992	NS	864	**854**	1.23	82	862	855	1.04	88	859	**854**	0.59	87
GJ08T	4674	NS	4183	**4134**	1.17	417	4170	**4134**	0.86	431	4179	4152	1.08	435
GJ09T	4087	NS	3727	3684	1.39	452	3718	**3677**	1.12	492	3716	3678	1.06	485
GJ10T	4002	NS	3540	3502	1.58	444	3524	**3485**	1.11	472	3516	3492	0.88	467
GJ11T	3794	NS	3428	**3390**	1.12	445	3421	**3390**	0.92	469	3432	3409	1.23	464
Tot.	22085		19977	19780		2243	19915	19745		2382	19921	19789		2371
Avg.	2008				0.93	204			0.63	217			0.61	216
< PB			11				11				11			
#B	0		7				10				6			

Table 21: Nagy Salhi MDMVRPB problems with 10% backhaul customers.

	Best known		Std. Removals				6R - no learning				6R - normal learning			
	cost	Reference	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
GJ01Q	666	NS	529	**528**	0.04	32	528	**528**	0.00	36	528	**528**	0.00	38
GJ02Q	550	NS	451	**450**	0.34	38	451	**450**	0.27	43	451	**450**	0.39	44
GJ03Q	670	NS	607	**605**	0.26	64	608	**605**	0.40	71	607	**605**	0.36	72
GJ04Q	1168	NS	879	876	0.47	87	876	**875**	0.13	94	880	876	0.55	95
GJ05Q	828	NS	705	**700**	0.72	124	705	702	0.65	133	706	703	0.83	134
GJ06Q	978	NS	805	**794**	1.39	92	800	**794**	0.78	100	799	**794**	0.60	100
GJ07Q	940	NS	808	803	0.69	89	807	803	0.51	94	806	**802**	0.45	95
GJ08Q	4877	NS	3826	3799	1.72	449	3810	3774	1.29	479	3792	**3762**	0.80	478
GJ09Q	4087	NS	3433	3391	2.31	482	3393	**3355**	1.13	535	3394	3362	1.15	535
GJ10Q	3931	NS	3294	3259	1.61	477	3267	3245	0.79	513	3276	**3242**	1.04	510
GJ11Q	3840	NS	3191	3171	1.15	472	3192	3165	1.17	511	3189	**3155**	1.10	505
Tot.	22535		18528	18375		2407	18437	18296		2609	18428	18279		2608
Avg.	2049				0.97	219			0.65	237			0.66	237
< PB			11				11				11			
#B	0		5				6				8			

Table 22: Nagy Salhi MDMVRPB problems with 25% backhaul customers.

	Best known		Std. Removals				6R - no learning				6R - normal learning			
	cost	Reference	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
GJ01H	619	NS	499	**499**	0.06	36	499	**499**	0.04	40	499	**499**	0.00	42
GJ02H	562	NS	440	**440**	0.00	44	440	**440**	0.00	51	440	**440**	0.00	53
GJ03H	662	NS	584	**581**	0.60	73	583	**581**	0.40	81	583	**581**	0.35	82
GJ04H	1055	NS	795	**789**	0.73	102	797	790	0.91	112	796	790	0.84	114
GJ05H	853	NS	681	**678**	0.50	154	680	**678**	0.28	168	680	**678**	0.27	171
GJ06H	1034	NS	753	748	1.06	106	751	747	0.80	116	751	**745**	0.91	118
GJ07H	932	NS	739	**733**	0.88	107	734	**733**	0.23	117	735	**733**	0.29	113
GJ08H	5188	NS	3391	3370	1.92	530	3373	**3327**	1.38	581	3371	3342	1.31	577
GJ09H	4087	NS	3043	**3005**	1.27	582	3028	3006	0.78	646	3027	3008	0.75	650
GJ10H	4041	NS	2961	2931	1.16	547	2963	2930	1.21	644	2962	**2927**	1.19	637
GJ11H	3933	NS	2898	**2855**	1.49	557	2905	2880	1.74	609	2893	2859	1.33	606
Tot.	22966		16785	16630		2841	16753	16611		3166	16738	16601		3163
Avg.	2088				0.88	258			0.71	288			0.66	288
< PB			11				11				11			
#B	0		8				6				7			

Table 23: Nagy Salhi MDMVRPB problems with 50% backhaul customers.

	Best known					Std. Removals				6R - no learning				6R - normal learning			
	% BH	n	m	cost	ref	avg. #veh.	best sol.	best #veh.	avg. time (s)	avg. #veh.	best sol.	best #veh.	avg. time (s)	avg. #veh.	best sol.	best #veh.	avg. time (s)
BHR1DO.10	10%	250	49	5085	TPS	46.0	4848.2	46	556	46.0	4844.8	46	571	46.0	**4843.9**	**46**	586
BHR1DO.30	20%	250	48	5243	TPS	45.0	5074.2	45	502	45.0	**5062.7**	**45**	512	45.0	5066.9	45	528
BHR1DO.50	50%	250	52	5403.1	TPS	49.0	5122.6	49	508	49.0	**5107.1**	**49**	531	49.0	5113.7	49	541
BHR1UP.10	10%	250	39	4278.6	TPS	32.0	3942.3	32	514	31.8	**4056.9**	**31**	503	32.0	3943.1	32	530
BHR1UP.30	30%	250	41	4715.2	TPS	35.0	4448.9	35	475	35.0	4427.8	35	474	34.8	**4549.7**	**34**	488
BHR1UP.50	50%	250	43	4921.4	TPS	36.0	4443.7	36	465	35.6	**4618.4**	**35**	473	36.0	4442.2	36	476
BHRC1DO.10	10%	250	39	4613.4	TPS	33.0	4116.8	33	506	32.8	4310.4	32	500	32.6	**4211.6**	**32**	519
BHRC1DO.30	20%	250	41	4852.2	TPS	34.4	**4506.3**	**34**	466	34.2	4534.4	34	466	34.2	4526.2	34	478
BHRC1DO.50	50%	250	41	4329.4	TPS	35.0	4500.2	35	456	34.4	**4513.9**	**34**	458	34.6	4589.6	34	463
BHRC1UP.10	10%	250	40	4445.8	TPS	33.4	4160.9	33	497	33.0	4137.0	33	488	33.4	**4105.3**	**33**	506
BHRC1UP.30	30%	250	43	4722.4	TPS	36.0	4485.2	36	466	35.0	**4538.0**	**35**	459	35.4	4555.8	35	469
BHRC1UP.50	50%	250	41	4899.4	TPS	35.6	4605.8	35	455	35.6	4558.5	35	464	35.2	**4550.2**	**35**	464
Tot.			517	57509		450.4	54255.1	449	5868	447.4	54710.0	444	5899	448.2	54498.3	445	6050
Avg.									489				492				504
< PB							12				12				12		
#B			0				1				6				5		

Table 24: Thangiah et al. 250 customer VRPBTW instances. The previously best known results have been found in [38] (TPS).

	Best known					Std. Removals				6R - no learning				6R - normal learning			
	% BH	n	m	cost	ref	avg. #veh.	best sol.	best #veh.	avg. time (s)	avg. #veh.	best sol.	best #veh.	avg. time (s)	avg. #veh.	best sol.	best #veh.	avg. time (s)
BHR1DO.10	10%	500	67	7620.4	TPS	58.4	6899.9	58	1726	58.0	**6860.2**	**58**	1691	58.0	6868.0	58	1763
BHR1DO.30	20%	500	69	9020.2	TPS	59.4	7320.8	59	1555	58.8	**7337.2**	**58**	1557	59.0	7262.3	59	1595
BHR1DO.50	50%	500	76	8376.5	TPS	61.8	7342.7	61	1554	61.0	7342.4	61	1575	60.8	**7294.7**	**60**	1584
BHR1UP.10	10%	500	64	7267.2	TPS	55.0	6776.6	54	1660	55.0	**6702.7**	**54**	1378	54.6	6784.7	54	1692
BHR1UP.30	30%	500	73	7926.6	TPS	57.8	7243.0	57	1533	57.8	7055.0	57	1679	57.6	**6991.0**	**57**	1566
BHR1UP.50	50%	500	68	8043.7	TPS	59.4	7119.1	59	1500	59.0	7126.2	59	1741	58.6	**7217.3**	**58**	1548
BHRC1DO.10	10%	500	61	7099.4	TPS	52.2	6362.6	52	1652	52.2	6346.8	52	1814	52.2	**6313.3**	**52**	1658
BHRC1DO.30	20%	500	63	7707.1	TPS	54.8	6959.3	54	1511	54.8	6889.0	54	1703	54.4	**6813.6**	**54**	1530
BHRC1DO.50	50%	500	65	7771.6	TPS	55.0	6983.7	54	1503	54.8	6914.5	54	1727	54.4	**6896.5**	**54**	1520
BHRC1UP.10	10%	500	63	7209.4	TPS	55.8	6493.5	55	1584	55.2	6483.4	55	1622	55.2	**6464.1**	**55**	1591
BHRC1UP.30	30%	500	63	7967.1	TPS	58.0	7030.2	57	1476	58.0	6918.8	58	1628	58.0	**7028.3**	**57**	1500
BHRC1UP.50	50%	500	67	8135.1	TPS	57.4	6965.8	57	1486	56.6	**6969.6**	**56**	1701	57.2	6862.3	57	1296
Tot.			799	94144		685.0	83497.3	677	18742	681.2	82945.7	676	19815	680.0	82796.0	675	18843
Avg.									1562				1651				1570
< PB							12				12				12		
#B			0				0				4				8		

Table 25: Thangiah et al. 500 customer VRPBTW instances. The previously best known results are the solutions found by Thangiah et al. (TPS) [38].

	Best known			Std. Removals						6R - no learning						6R - normal learning					
	veh.	cost	Reference	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)
MC201	5	763.88	ZC	774.59	4.0	**766.82**	**4**	1.01	140	769.96	4.0	**766.82**	**4**	0.41	141	766.82	4.0	**766.82**	**4**	0.00	144
MC202	4	1186.24	ZC	736.76	4.0	**732.93**	**4**	0.52	165	734.85	4.0	**732.93**	**4**	0.26	171	737.51	4.0	**732.93**	**4**	0.63	165
MC203	4	1096.31	ZC	710.14	4.0	705.86	4	0.80	171	708.68	4.0	707.75	4	0.60	177	708.48	4.0	**704.49**	**4**	0.57	174
MC204	4	885.73	ZC	677.75	4.0	**676.18**	**4**	0.23	188	679.06	4.0	**676.18**	**4**	0.43	193	678.54	4.0	**676.18**	**4**	0.35	189
MC205	5	781.7	ZC	754.81	4.0	**748.34**	**4**	0.86	152	755.72	4.0	751.96	4	0.99	153	758.83	4.0	751.96	4	1.40	152
MC206	5	860.74	ZC	750.16	4.0	748.17	4	0.41	157	748.33	4.0	**747.08**	**4**	0.17	158	748.09	4.0	**747.08**	**4**	0.14	157
MC207	5	792.96	ZC	745.38	4.0	**737.39**	**4**	1.08	161	745.43	4.0	**737.39**	**4**	1.09	164	745.57	4.0	738.70	4	1.11	162
MC208	5	859.92	ZC	736.19	4.0	**735.17**	**4**	0.14	161	741.69	4.0	738.70	4	0.89	164	742.76	4.0	738.70	4	1.03	162
Tot.	37	7227		5885.79	32.00	5850.87	32		1295	5883.72	32.00	5858.82	32		1320	5886.63	32.00	5856.87	32		1305
Avg.	5							0.63	162					0.60	165					0.65	163
< PB						8						8						8			
#B		0				6						5						5			

Table 26: Kontoravdis and Bard's MVRPBTW instances. C-type problems. The previously best known results are the solutions found by Zhong and Cole (ZC) [48]. Kontoravdis and Bard [23] do not give detailed information about their solutions.

	Best known			Std. Removals						6R - no learning						6R - normal learning					
	veh.	cost	Reference	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)
MR201	4	1388.73	ZC	1272.20	4.0	1260.48	4	1.27	157	1263.64	4.0	**1256.31**	**4**	0.58	165	1261.90	4.0	**1256.31**	**4**	0.45	160
MR202	4	1198.99	ZC	1101.54	4.0	1092.01	4	1.39	359	1092.08	4.0	**1086.46**	**4**	0.52	371	1092.36	4.0	**1086.46**	**4**	0.54	362
MR203	4	988.82	ZC	913.57	4.0	**894.54**	**4**	2.13	387	900.08	4.0	896.14	4	0.62	383	899.59	4.0	896.14	4	0.56	374
MR204	4	858.32	ZC	739.43	4.0	737.51	4	0.36	419	737.87	4.0	737.51	4	0.15	436	738.63	4.0	**736.75**	**4**	0.25	432
MR205	4	1172.53	ZC	994.25	4.0	**974.26**	**4**	2.05	351	994.86	4.0	**974.26**	**4**	2.11	367	989.36	4.0	**974.26**	**4**	1.55	353
MR206	4	979.5	ZC	910.42	4.0	897.03	4	1.83	386	896.47	4.0	894.05	4	0.27	397	894.25	4.0	**894.04**	**4**	0.02	388
MR207	4	912.69	ZC	811.92	4.0	**800.79**	**4**	1.39	421	800.79	4.0	**800.79**	**4**	0.00	426	800.79	4.0	**800.79**	**4**	0.00	422
MR208	4	764.52	ZC	722.34	4.0	719.12	4	0.85	412	719.05	4.0	**716.28**	**4**	0.39	435	718.91	4.0	**716.28**	**4**	0.37	431
MR209	4	978.82	ZC	894.18	4.0	**879.63**	**4**	1.65	354	886.97	4.0	**879.63**	**4**	0.83	371	893.49	4.0	881.60	4	1.58	361
MR210	4	1061.36	ZC	936.45	4.0	930.92	4	1.29	361	929.49	4.0	**924.56**	**4**	0.53	377	928.71	4.0	**924.56**	**4**	0.45	369
MR211	4	878.81	ZC	767.51	4.0	763.54	4	0.58	376	770.42	4.0	**763.09**	**4**	0.96	400	772.21	4.0	765.03	4	1.19	394
Tot.	44	11183		10063.80	44.00	9949.83	44		3983	9991.73	44.00	9929.07	44		4128	9990.20	44.00	9932.21	44		4046
Avg.	4							1.34	362					0.63	375					0.63	368
< PB						11						11						11			
#B		0				4						8						8			

Table 27: Kontoravdis and Bard's MVRPBTW instances. R-type problems.

	Best known			Std. Removals						6R - no learning						6R - normal learning					
	veh.	cost	Reference	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)
MRC201	5	1498.9	ZC	1370.16	5.0	**1346.30**	**5**	1.77	234	1357.26	5.0	1355.42	5	0.81	238	1356.84	5.0	1355.63	5	0.78	236
MRC202	4	1539.41	ZC	1263.92	4.0	**1230.24**	**4**	2.74	171	1261.06	4.0	1241.77	4	2.51	174	1250.88	4.0	**1230.24**	**4**	1.68	171
MRC203	4	1303.48	ZC	1020.29	4.0	997.06	4	2.48	177	1004.33	4.0	**995.63**	**4**	0.87	175	999.79	4.0	**995.63**	**4**	0.42	176
MRC204	4	932.48	ZC	843.99	4.0	**833.60**	**4**	1.25	187	844.08	4.0	835.13	4	1.26	188	846.01	4.0	836.89	4	1.49	187
MRC205	4	1632.04	ZC	1461.20	4.0	1417.14	4	3.30	168	1449.27	4.0	1419.07	4	2.46	166	1452.19	4.0	**1414.52**	**4**	2.66	160
MRC206	4	1433.43	ZC	1286.51	4.0	**1231.52**	**4**	4.47	168	1277.35	4.0	1249.48	4	3.72	170	1291.85	4.0	1254.51	4	4.90	166
MRC207	4	1217.2	ZC	1119.23	4.0	1096.06	4	3.31	175	1109.06	4.0	1084.81	4	2.37	174	1101.21	4.0	**1083.33**	**4**	1.65	169
MRC208	4	1085.57	ZC	875.86	4.0	**847.46**	**4**	3.35	180	863.90	4.0	852.25	4	1.94	182	851.50	4.0	849.30	4	0.48	175
Tot.	33	10643		9241.15	33.00	8999.36	33		1461	9166.31	33.00	9033.57	33		1467	9150.29	33.00	9020.05	33		1441
Avg.	4							2.83	183					1.99	183					1.76	180
< PB						8						8						8			
#B		0				5						1						4			

Table 28: Kontoravdis and Bard's MVRPBTW instances. RC-type problems.

References

[1] R.K. Ahuja, O. Ergun, J.B. Orlin, A.P Punnen *A survey of very large scale neighborhood search techniques*, Discrete Applied Mathematics **123** 75–102 (2002).

[2] E.Angelelli, R. Mansini, *The vehicle routing problem with time windows and simultaneous pick-up and delivery*, in *Quantitative Approaches to Distribution Logistics and Supply Chain Management*, (edited by A. Klose, M. G. Speranza, L. N. Van Wassenhove), Springer-Verlag, 249–267 (2002)

[3] S. Anily, *The vehicle-routing problem with delivery and back-haul options*, Naval Research Logistics **43** 415–434 (1996).

[4] R. Battiti, G. Tecchiolli, *The reactive tabu search*, ORSA Journal of Computing **6** 126–140 (1994).

[5] D.O. Casco, B.L. Golden, E.A. Wasil, *Vehicle routing with backhauls: models, algorithms and case studies*, in *Vehicle Routing: Methods and Studies* (Edited by B. Golden and A. Assad), North-Holland, Amsterdam 127–147 (1988).

[6] J.-F. Cordeau, G. Laporte, A. Mercier, *A unified tabu search heuristic for vehicle routing problems with time windows*, Journal of the Operational Research Society, **52** 928–936 (2001).

[7] J. Crispim, J. Brandao, *Reactive tabu search and variable neighbourhood descent applied to the vehicle routing problem with backhauls*, MIC'2001 4th Metaheuristic International Conference, Porto, Portugal July 16–20 (2001).

[8] G. Desaulniers, J. Desrosiers, A. Ercmann, M.M. Solomon, F. Soumis, *VRP with pickup and delivery*, In P. Toth and D. Vigo (eds.): The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications **9**, SIAM, Philadelphia, 225-242, (2002).

	Best known			Std. Removals				6R - no learning				6R - normal learning			
	n	cost	reference	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
Min	22	**88**	H	88.3	**88.0**	0.34	37	88.5	**88.0**	0.57	44	88.5	**88.0**	0.57	50
SCA3-0	50	689	D	640.6	640.5	0.71	173	641.1	640.5	0.79	173	638.3	**636.1**	0.35	232
SCA3-1	50	765.6	D	698.4	**697.8**	0.08	170	698.8	**697.8**	0.14	173	697.8	**697.8**	0.00	218
SCA3-2	50	742.8	D	659.3	**659.3**	0.00	161	660.2	**659.3**	0.14	160	659.3	**659.3**	0.00	203
SCA3-3	50	737.2	D	681.4	**680.6**	0.12	182	682.7	681.3	0.31	179	681.4	**680.6**	0.11	241
SCA3-4	50	747.1	D	691.2	**690.5**	0.11	160	693.1	**690.5**	0.38	166	691.0	**690.5**	0.08	208
SCA3-5	50	784.4	D	662.2	**659.9**	0.34	178	660.5	**659.9**	0.10	179	659.9	**659.9**	0.00	226
SCA3-6	50	720.4	D	651.3	**651.1**	0.04	179	652.1	**651.1**	0.15	171	651.3	**651.1**	0.04	233
SCA3-7	50	707.9	D	667.9	**666.1**	0.27	169	667.0	**666.1**	0.13	162	667.0	**666.1**	0.13	206
SCA3-8	50	807.2	D	721.3	**719.5**	0.26	167	719.5	**719.5**	0.00	157	719.5	**719.5**	0.00	190
SCA3-9	50	764.1	D	681.0	**681.0**	0.01	171	681.0	**681.0**	0.01	167	681.0	**681.0**	0.00	220
SCA8-0	50	1132.9	D	991.1	982.2	1.63	82	993.2	987.9	1.85	94	986.3	**975.1**	1.15	98
SCA8-1	50	1150.9	D	1083.1	1072.8	2.92	82	1082.6	1068.8	2.87	94	1066.5	**1052.4**	1.35	95
SCA8-2	50	1100.8	D	1046.3	**1039.6**	0.64	83	1049.9	1044.5	0.99	87	1049.2	1044.5	0.92	94
SCA8-3	50	1115.6	D	1016.5	1007.8	2.49	85	1012.5	**991.8**	2.08	91	1006.3	999.1	1.45	94
SCA8-4	50	1235.4	D	1067.4	**1065.5**	0.18	84	1067.0	**1065.5**	0.14	87	1065.6	**1065.5**	0.01	93
SCA8-5	50	1231.6	D	1052.8	1039.6	2.50	84	1047.9	1040.4	2.02	89	1039.9	**1027.1**	1.24	96
SCA8-6	50	1062.5	D	996.2	986.0	2.44	82	987.5	**972.5**	1.54	93	983.5	977.0	1.14	94
SCA8-7	50	1217.4	D	1067.1	1062.2	0.57	82	1068.3	1063.2	0.69	88	1065.8	**1061.0**	0.45	92
SCA8-8	50	1231.6	D	1086.4	**1071.2**	1.42	85	1084.3	1077.7	1.22	93	1078.8	**1071.2**	0.71	98
SCA8-9	50	1185.6	D	1077.0	1067.3	1.55	82	1068.8	**1060.5**	0.79	86	1064.7	**1060.5**	0.40	92
CON3-0	50	672.4	D	623.4	617.6	1.11	173	621.5	**616.5**	0.81	171	619.0	**616.5**	0.40	215
CON3-1	50	570.6	D	558.1	**554.5**	0.65	190	555.5	**554.5**	0.18	190	554.5	**554.5**	0.00	245
CON3-2	50	534.8	D	522.3	**521.4**	0.18	176	523.0	**521.4**	0.32	177	521.6	**521.4**	0.05	232
CON3-3	50	656.9	D	591.2	**591.2**	0.00	185	591.2	**591.2**	0.00	177	591.2	**591.2**	0.00	231
CON3-4	50	640.2	D	591.7	**588.8**	0.49	187	590.5	**588.8**	0.29	173	590.0	**588.8**	0.21	221
CON3-5	50	604.7	D	566.3	**563.7**	0.47	181	567.3	**563.7**	0.64	179	564.4	**563.7**	0.12	209
CON3-6	50	521.3	D	501.6	**499.1**	0.51	195	503.0	501.8	0.78	180	501.9	500.8	0.57	225
CON3-7	50	602.8	D	579.7	577.5	0.56	178	584.1	578.4	1.32	181	579.5	**576.5**	0.53	227
CON3-8	50	556.2	D	523.5	**523.1**	0.08	186	523.7	**523.1**	0.12	174	523.5	**523.1**	0.08	237
CON3-9	50	612.8	D	585.9	**578.2**	1.32	175	587.4	**578.2**	1.58	163	588.2	586.4	1.71	207
CON8-0	50	967.3	D	867.8	**857.2**	1.24	86	867.7	858.0	1.22	87	860.9	**857.2**	0.43	94
CON8-1	50	828.7	D	761.0	**740.9**	2.72	85	754.2	741.7	1.81	92	750.5	**740.9**	1.30	94
CON8-2	50	770.2	D	731.3	719.3	2.14	85	728.8	718.3	1.79	93	721.4	**716.0**	0.75	94
CON8-3	50	906.7	D	827.9	822.9	2.07	88	816.7	**811.1**	0.69	91	813.7	**811.1**	0.33	98
CON8-4	50	876.8	D	779.1	**772.3**	0.89	88	779.3	**772.3**	0.91	87	774.3	**772.3**	0.27	95
CON8-5	50	866.9	D	773.9	763.1	2.41	85	772.2	763.1	2.19	92	766.5	**755.7**	1.44	94
CON8-6	50	749.1	D	717.0	705.8	3.46	88	712.5	696.9	2.81	95	707.9	**693.1**	2.14	96
CON8-7	50	929.8	D	844.8	831.5	3.69	86	843.1	818.0	3.48	94	833.1	**814.8**	2.24	94
CON8-8	50	833.1	D	781.2	774.1	0.93	87	781.3	775.9	0.94	88	778.8	**774.0**	0.63	94
CON8-9	50	877.3	D	813.3	812.0	0.49	86	814.5	812.0	0.64	91	813.0	**809.3**	0.46	92
Tot.		33797		30867.6	30642.7		5267	30823.9	30592.8		5308	30695.7	30530.3		6368
Avg.		824				1.07	128			0.96	129			0.58	155
< PB					40				40				40		
#B		1			24				23				36		

Table 29: The first problem in the table is Min's 21 customer problem which was solved to optimality by Halse (H) [19]. The rest of the problems were proposed by Dethloff (D) [10].

	Best known			Std. Removals				6R - no learning				6R - normal learning			
	n	cost	reference	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
SN1X	50	501	D	475	**467**	1.75	171	472	**467**	1.22	190	473	**467**	1.27	221
SN2X	75	782	D	718	**702**	2.40	255	722	709	2.84	271	719	704	2.46	294
SN3X	100	847	D	739	**727**	1.56	768	746	731	2.54	695	741	731	1.93	863
SN4X	150	1050	D	919	894	4.75	1345	903	**877**	2.95	1459	893	879	1.79	1676
SN5X	199	1348	D	1132	**1108**	2.13	2057	1162	1138	4.83	2217	1130	**1108**	1.99	2340
SN6X	50	584	D	559	**559**	0.08	97	559	**559**	0.08	98	559	**559**	0.08	113
SN7X	75	961	D	918	905	1.82	154	917	903	1.72	158	910	**901**	0.95	167
SN8X	100	923	NS	872	**866**	0.69	384	872	**866**	0.74	367	868	**866**	0.29	413
SN9X	150	1215	NS	1239	1221	3.54	703	1235	**1197**	3.17	726	1228	1205	2.57	765
SN10X	199	1571	D	1526	1494	4.42	1136	1522	1490	4.13	1214	1501	**1462**	2.71	1275
SN11X	120	959	D	907	875	8.33	1725	921	875	10.05	1410	901	**837**	7.70	1821
SN12X	100	804	D	698	688	2.11	628	692	**683**	1.32	574	692	685	1.29	684
SN13X	120	**1576**	D	1637	1595	3.90	494	1601	1591	1.57	539	1593	1578	1.09	563
SN14X	100	871	D	904	876	4.69	354	896	**863**	3.75	367	897	885	3.87	387
Tot.		13992		13242	12976		10270	13219	12947		10286	13105	12866		11585
Avg.		999				3.01	734			2.92	735			2.14	827
< PB					11				13				12		
#B		1			6				7				7		

Table 30: Nagy and Salhi's VRPSDP instances. X-type problems.

	Best known			Std. Removals				6R - no learning				6R - normal learning			
	n	cost	reference	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
SN1Y	50	501	D	470	**467**	0.69	194	471	**467**	0.88	192	469	**467**	0.53	235
SN2Y	75	782	D	692	**685**	1.04	315	704	691	2.88	268	694	**685**	1.34	331
SN3Y	100	847	D	743	**734**	1.20	632	751	742	2.28	625	747	738	1.71	708
SN4Y	150	1050	D	868	**854**	1.57	1866	876	856	2.56	1487	881	876	3.08	1788
SN5Y	199	1348	D	1158	**1131**	2.37	2030	1186	1132	4.86	2106	1169	1146	3.31	2177
SN6Y	50	584	D	560	**559**	0.21	93	562	**559**	0.63	96	560	**559**	0.20	101
SN7Y	75	961	D	987	969	3.73	158	1008	979	5.92	163	993	**952**	4.38	166
SN8Y	100	923	NS	896	880	2.63	361	916	894	4.85	362	895	**873**	2.43	398
SN9Y	150	**1215**	NS	1282	1267	5.55	732	1286	1256	5.83	720	1288	1271	6.03	757
SN10Y	199	**1527**	NS	1597	1567	4.57	1207	1596	1573	4.54	1195	1591	1552	4.16	1255
SN11Y	120	1070	D	972	938	5.71	1193	980	956	6.54	1154	951	**920**	3.42	1376
SN12Y	100	825	D	683	**673**	1.58	531	689	686	2.39	506	684	675	1.65	539
SN13Y	120	**1576**	D	1771	1726	12.38	531	1629	1612	3.34	538	1613	1602	2.33	547
Tot.		13209		12680	12451		9844	12654	12403		9412	12534	12315		10378
Avg.		944				3.33	703			3.65	672			2.66	741
< PB					9				9				10		
#B		3			7				2				6		

Table 31: Nagy and Salhi's VRPSDP instances. Y-type problems.

[9] J. Dethloff, *Relation between vehicle routing problems: an insertion heuristic for the vehicle routing problem with simultaneous delivery and pick-up applied to the vehicle routing problem with backhauls*, Journal of the Operational Research Society **53** 115–118 (2002).

[10] J. Dethloff, *Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up*, OR Spektrum **23** 79-96 (2001).

[11] J.J. Dongarra, *Performance of various computers using standard linear equation software*, University of Tennessee Computer Science Technical Report, CS-89-85, (2004).

[12] C. Duhamel, J.-Y. Potvin, J.-M. Rousseau, *A tabu search heuristic for the vehicle routing problem with backhauls and time windows*, Transportation Science **31** 49–59 (1997).

[13] S. Gelinas, M. Desrochers, J. Desrosiers, M.M. Solomon., *A new branching strategy for time constrained routing problems with application to backhauling*, Annals of Operations Research **61** 91–109 (1995).

[14] M. Gendreau, G. Laporte, D. Vigo, *Heuristics for the traveling salesman problem with pickup and delivery*, Computers & Operations Research **26**, 699–714 (1999).

[15] H. Ghaziri, I.H. Osman, *A neural network algorithm for the traveling salesman problem with backhauls*, Computers & Industrial Engineering **44**, 267–281 (2003).

[16] M. Goetschalckx, C. Jacobs-Blecha, *The vehicle routing problem with backhauls*, European Journal of Operational Research **42** 39–51 (1989).

[17] Ø. Halskau; I. Gribkovskaia; K.N.B. Myklebost. *Models for pick-up and deliveries from depots with lasso solutions*. Proceedings of the 13th Annual Conference on Logistics Research - NOFOMA 2001, Collaboration in logistics : Connecting Islands using Information Technology. Reykjavik, Iceland, 2001-06-14 - 2001-06-15. Chalmers University of Technology, Göteborg, Sweden. 279–293 (2001).

[18] P. Hansen, N. Mladenovic, *An introduction to variable neighborhood search*, in: S. Voss et. al. (ed) Meta-Heuristics, Advances and Trends in Local Search Paradigms for Optimization, Kluwer, Boston, 433–458 (1999).

[19] K. Halse, *Modeling and solving complex vehicle routing problems*, PhD thesis, Institute of Mathematical Statistics and Operations Research (IMSOR), Technical University of Denmark (1992).

[20] T. Hasama, H. Kokubugata, H. Kawashima, *A heuristic approach based on the string model to solve vehicle routing problem with backhauls*, Proceedings of the 5th World Congress on Intelligent Transport Systems (ITS), Seoul, 1998.

[21] A. Hertz, E.D. Taillard, D. de Werra *A tutorial on tabu search*, in: E.H.L. Aarts and J. K. Lenstra, Local search in combinatorial optimization, Wiley, 121–136 (1997)

[22] C. Jacobs-Blecha, M. Goetschalckx, *The vehicle routing problem with backhauls: properties and solution algorithms*, Technical Report, 1992-1998, Georgia Tech Research Corporation.

[23] G. Kontoravdis, J.F. Bard, *A GRASP for the vehicle routing problem with time windows*, ORSA Journal on Computing **7** 10–23 (1995).

[24] J.B. Kruskal, *On the shortest spanning subtree of a graph and the traveling salesman problem*, Proceedings of the American Mathematical Society **7** 48–50 (1956).

[25] H. Min, *The multiple vehicle routing problem with simultaneous delivery and pickup*, Transportation Research Part A **23** 377–386 (1989).

[26] A. Mingozzi, S. Giorgi, R. Baldacci, *An exact method for the vehicle routing problem with backhauls*, Transportation Science **33**, 315–329 (1999).

[27] G. Nagy, S. Salhi, *Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries*, Working Paper no. 42, Canterbury Business School, 2003.

[28] I.H. Osman, N.A. Wassan, *A reactive tabu search meta-heuristic for the vehicle routing problem with backhauls*, Journal of Scheduling **5** 263–285 (2002).

[29] J.Y. Potvin, J.-M. Rousseau, *A parallel route building algorithm for the vehicle routing and scheduling problem with time windows*, European Journal of Operational Research **66** 331–340 (1993).

[30] M. Reimann, Doerner K., Hartl R.F., *Insertion based ants for vehicle routing problems with backhauls and time windows*, LNCS 2463 135–148 (2002).

[31] S. Ropke, *A local Search heuristic for the pickup and delivery problem with time windows*, Working paper, DIKU, University of Copenhagen (2003).

[32] S. Salhi, G. Nagy, *A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling*, Journal of the Operational Research Society (1999) 50, 1034-1042.

[33] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, G. Dueck, *Record breaking optimization results – using the ruin & recreate principle*, J. Comp. Phys. **159**, 139-171 (2000).

[34] M. Sigurd, D. Pisinger, M. Sig, *The pickup and delivery problem with time windows and precendences*, Transportation Science **38** 197–209 (2004).

[35] P. Shaw, *Using constraint programming and local search methods to solve vehicle routing problems*, Proceedings CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming) (1998).

[36] H. Süral, J.H. Bookbinder, *The single-vehicle routing problem with unrestricted backhauls*, Networks **41**, 127–136 (2003).

[37] M.M. Solomon, *Algorithms for the vehicle routing and scheduling problems with time window constraints* Operations Research, **35** 254–265, (1987).

[38] S.R. Thangiah, J.-Y. Potvin, Sun T., *Heuristic approaches to vehicle routing with backhauls and time windows*, Computers & Operations Research **23** 1043–1057 (1996).

[39] F.A. Tillman, T. M. Cain *An upper bound algorithm for the single and multiple terminal delivery problem* Management Science **18** 664-682 (1972).

[40] P. Toth, D. Vigo, *A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls*, European Journal of Operational Research **113** 528–543 (1999).

[41] P. Toth, D. Vigo, *An exact algorithm for the vehicle routing problem with backhauls*, Transportation Science **31** 372-285 (1997).

[42] P. Toth, D. Vigo, *VRP with backhauls*, In P. Toth and D. Vigo (eds.): The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications **9**, SIAM, Philadelphia, 195-221, (2002).

[43] A. Wade, S. Salhi, *An ant system algorithm for the mixed vehicle routing problem with backhauls*, in M.G.C. Resende and J.P. de Sousa (eds.): Metaheuristics: Computer Decision-Making, Chapter 33, 699-719, Kluwer (2003).

[44] A. Wade, S. Salhi, *An ant system algorithm for the vehicle routing problem with backhauls*, MIC'2001 **-** 4th Metaheursistic International Conference.

[45] A.C. Wade, S. Salhi, *An investigation into a new class of vehicle routing problem with backhauls*, Omega **30** 497–487 (2002).

[46] Web page: www.diku.dk/~sropke

[47] Y. Zhong, M.H. Cole, *A simple approach to linehaul-backhaul problems: a guided local search approach for the vehicle routing problem*, Technical Report, Department of Industrial Engineering, University of Arkansas, USA, 2001.

[48] Y. Zhong, M.H. Cole, *A vehicle routing problem with backhauls and time windows: a guided local search solution*, Transportation Research Part E, Article in press (2004).

Chapter 6

A general heuristic for vehicle routing problems

A general heuristic for vehicle routing problems

David Pisinger *
Stefan Ropke *

Abstract

We present a unified heuristic which is able to solve five different variants of the vehicle routing problem: the vehicle routing problem with time windows (VRPTW), the capacitated vehicle routing problem (CVRP), the multi-depot vehicle routing problem (MDVRP), the site dependent vehicle routing problem (SDVRP) and the open vehicle routing problem (OVRP).

All problem variants are transformed into a rich pickup and delivery model and solved using the Adaptive Large Neighborhood Search (ALNS) framework presented in Ropke and Pisinger (2004). The ALNS framework is an extension of the Large Neighborhood Search framework by Shaw (1998) with an adaptive layer. This layer adaptively chooses among a number of insertion and removal heuristics to intensify and diversify the search. The presented approach has a number of advantages: it provides solutions of very high quality, the algorithm is robust, and to some extent self-calibrating. Moreover, the unified model allows the dispatcher to mix various variants of VRP problems for individual customers or vehicles.

As we believe that the ALNS framework can be applied to a large number of tightly constrained optimization problems, a general description of the framework is given, and it is discussed how the various components can be designed in a particular setting.

The paper is concluded with a computational study, in which the five different variants of the vehicle routing problem are considered on standard benchmark tests from the literature. The outcome of the tests is promising as the algorithm is able to improve 183 best known solutions out of 486 benchmark tests. The heuristic has also shown promising results for a large class of vehicle routing problems with backhauls as demonstrated in Ropke and Pisinger (2005).

Keywords: metaheuristics, large neighborhood search, vehicle routing problem

1 Introduction

Most scientific papers in the area of heuristic solution methods for vehicle routing problems target a specific vehicle routing problem, e.g. vehicle routing problems with time windows (VRPTW). In such papers a heuristic is designed, implemented and fine-tuned to fit this particular problem type. Only a few papers (see e.g. Cordeau et al. [17, 19]) consider heuristics that “out-of-the-box” can be used to solve several problem types. We believe that general vehicle routing heuristics are an important research area as such heuristics are needed for real life problems, in which the transportation needs of different companies often are different and thus call for various types of vehicle routing problems.

The heuristic in this paper is applied to five different problems: the vehicle routing problem with time windows (VRPTW), the capacitated vehicle routing problem (CVRP), the multi-depot vehicle routing problem (MDVRP), the site dependent vehicle routing problem (SDVRP) and the open vehicle routing problem (OVRP). In the CVRP one has to deliver goods to a set of customers with known demands on minimum-cost vehicle routes originating and terminating at a depot. The vehicles are assumed to be homogeneous and having a certain capacity. In some versions of the CVRP one also has to obey a route duration constraint that limits the lengths of the feasible routes. The VRPTW extends the CVRP by associating

* DIKU - Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark. E-mail: {pisinger, sropke}@diku.dk

time windows with the customers. The time window defines an interval during which the customer must be visited. The OVRP is closely related to the CVRP, but contrary to the CVRP a route ends as soon as the last customer has been served as the vehicles do not need to return to the depot. The MDVRP extends the CVRP by allowing multiple depots. The SDVRP is another generalization of the CVRP in which one can specify that certain customers only can be served by a subset of the vehicles. Furthermore, vehicles do not need to have the same capacity in the SDVRP. In the CVRP, MDVRP and SDVRP one seeks to minimize the total traveled distance whereas in the OVRP and VRPTW, the first priority is to minimize the number of vehicles and minimizing the traveled distance is the second priority. The choice of objective is not an intrinsic feature of the problems, but just the tradition in the metaheuristic literature. Most exact methods and some metaheuristics for the VRPTW minimize total traveled distance instead of minimizing number of vehicles used.

All problem types are transformed to a rich pickup and delivery problem with time windows (RPDPTW) and are solved using the adaptive large neighborhood search (ALNS) framework introduced by [49, 50]. The heuristic presented in the two aforementioned papers has been reused, with some small improvements (summarized in section 5), to solve the five problem types considered in this paper.

In the RPDPTW we have a number of requests to be carried out by a given set of vehicles. Each request consists of picking up a quantity of goods at one location and delivering it to another location. The objective of the problem is to find a feasible set of routes for the vehicles so that all requests are serviced, and such that the overall travel distance is minimized. A feasible route of a vehicle must start at a given location, service a number of requests such that the capacity of the vehicle is not exceeded, and finally end at a given location. A pickup or delivery must take place within a given time window. Each request has an associated pickup precedence number, and a delivery precedence number. A vehicle must visit the locations in nondecreasing order of precedences (see e.g. Sigurd et al. [54] for various applications of precedence constraints). Since not all vehicles may be able to service all requests (e.g. due to their physical size or the absence of some cooling compartments) we need to ensure that every request is serviced by a given subset of vehicles. Between any two locations we have an associated, nonnegative distance and travel time. It is assumed that travel times satisfy the *triangle inequality*. This assumption implies that any removal of requests from a feasible route will keep the route feasible with respect to the imposed time windows.

The five vehicle routing problems considered in the present paper have all been intensively studied in the literature. The two best known problems are the VRPTW and the CVRP. The VRPTW has been the target of extensive research and almost any type of metaheuristic has been applied to the problem. For recent surveys on the state of the art in VRPTW research we recommend the survey by Cordeau et al [15] that describes both exact and heuristic methods, and the survey by Bräysy and Gendreau [8] that focuses on metaheuristics. It is hard to single out a few VRPTW metaheuristics as the number of proposed heuristics is huge, and no heuristic dominates all the other heuristics in all aspects. We would, however, like to mention the metaheuristic by Mester and Bräysy [42] as it has achieved outstanding results on larger VRPTW instances with between 200 and 1000 customers. For the smaller VRPTW instances like the Solomon data set, some of the best heuristics in terms of solution quality achieved are the Large Neighborhood Search by Bent and Van Hentenryck [2] and the Hybrid Genetic Algorithm by Homberger and Gehring [32].

Solving the VRPTW to optimality has also received much attention. The current state of the art exact methods are proposed by Kallehauge et al [35], Irnich and Villeneuve [34] and Chabrier [10], and all follow the branch-and-price framework. The two first mentioned approaches also strengthen the obtained lower bound by adding valid inequalities to the LP formulation. The size of the instances that consistently can be solved to optimality is rather limited as unsolved instances with 50 customers exist, but some large scale instances can be solved. For example, Kallehauge et al. [35] report that a 1000 customer instance has been solved. Solving problems of this size is only possible by current techniques if the instance has a certain structure and the time constraints are very tight. These observations justify the research into heuristics for the VRPTW as industrial routing problems demand robust algorithms for large-sized instances.

The CVRP literature is also vast. Classic heuristics for the problem have been surveyed by Laporte and Semet [38], and metaheuristics have been surveyed by Gendreau et al. [29] and more recently by Cordeau et al. [16]. CVRP heuristics have typically been tested on 14 instances containing between 50 and 199 customers. In the early '90s very good metaheuristics for the CVRP were developed such as parallel tabu search by Taillard [56]. Most of the solutions to the 14 classic instances found back then have still not been improved. More recently, some larger instances have been introduced containing between 240 and 1200 customers (Golden et al. [30] and Li et al. [40]). These new instances seem to have spurred a new interest into metaheuristics for the CVRP as indicated in the survey by Cordeau et al. [16].

Until recently, exact methods for the CVRP were dominated by branch-and-cut methods. One of the best branch-and-cut algorithms for the CVRP was developed by Lysgaard et al. [41]. Recent research results indicate that branch-and-cut-and-price algorithms are a more promising approach as shown by Fukasawa et al. [26]. For the CVRP, the largest problem that has been solved to optimality contains 135 customers.

The OVRP is a variant of the CVRP that has received less attention. The problem appears in various distribution problems, in which the vehicle simply stops after the last delivery. The problem was introduced by Sariklis and Powell [51] and they proposed a two-phase cluster first-route second heuristic. Recently, tabu search heuristics were proposed by Fu et al. [25] and Brandão [6].

Tabu search heuristics for the MDVRP have been proposed by Renaud et al. [48] and Cordeau et al. [17]. The last paper deserves special attention as it describes a general heuristic that also solves periodic vehicle routing problems (PVRP) and periodic traveling salesman problems. Earlier, Chao et al. [11] proposed a *record-to-record* improvement heuristic for the MDVRP.

The SDVRP was first studied by Nag et al. [43] who developed several simple heuristics for the problem. Chao et al. [12] developed a more advanced heuristic and constructed several new test instances. Cordeau and Laporte [18] showed that the problem could be seen as a special case of the PVRP and they presented computational results obtained by solving the problem using their PVRP tabu search heuristic.

The main contribution of this paper is to describe a general ALNS heuristic that is able to solve all the above variants of the VRP problem. The computational results are promising as the ALNS, for the large scale VRPTW instances suggested by Gehring and Homberger [27], on average use less vehicles compared to competing heuristics , and the method becomes even more attractive compared to other heuristics as the problem size increases. For the OVRP, MDVRP and SDVRP we are able to improve a large number of best known solutions. The ALNS heuristic is comparable to most recently proposed heuristics for the CVRP, but it is surpassed by the very best heuristic for the problem type.

Due to the promising results of ALNS, we give a general description of the paradigm to make it easier to adapt the framework to other problem types. Various strategies for designing construction and removal heuristics are discussed.

In Section 2 we give a formal mathematical definition of the RPDPTW and in Section 3 we describe how the considered problem variants are transformed into the RPDPTW. In Section 4 we give a general presentation of the ALNS algorithm forming the core of our solution approach. Section 5 describes how the general framework has been adapted to solve the RPDPTW. Section 6 presents a number of computational experiments which document that the proposed heuristic does not perform worse than state-of-the-art heuristics specialized to solve each problem variant. The paper is concluded in Section 7.

2 Formal problem definition

We now present a mathematical formulation of the RPDPTW problem. The mathematical model is used to describe the heuristic in details in later sections and to describe how the considered VRP variants are transformed to the RPDPTW.

Following the terminology of Desaulniers et al. [22], a problem instance of the pickup and delivery problem contains n requests and m vehicles. The problem is defined on a graph where $P = \{1, \ldots, n\}$ is the set of pickup nodes, and $D = \{n + 1, \ldots, 2n\}$ is the set of delivery nodes. Request i is represented by node i and $i + n$. $K = \{1, \ldots, m\}$ is the set of all vehicles. Let $P_k \subseteq P$ and $D_k \subseteq D$ be the set of pickups and deliveries that can be served by vehicle k. Since a request is serviced by the same vehicle we may assume that $i \in P_k \Leftrightarrow i + n \in D_k$, i.e. that both the pickup and delivery can be serviced by vehicle k. Define $N = P \cup D$ and $N_k = P_k \cup D_k$. Let $\tau_k = 2n + k,\ k \in K$ and $\tau'_k = 2n + m + k,\ k \in K$ be the nodes that represent the start and end terminals of vehicle k. The directed graph $G = (V, A)$ consists of the nodes $V = N \cup \{\tau_1, \ldots, \tau_m\} \cup \{\tau'_1, \ldots, \tau'_m\}$ and the arcs $A = V \times V$. For each vehicle we have a subgraph $G_k = (V_k, A_k)$, where $V_k = N_k \cup \{\tau_k\} \cup \{\tau'_k\}$ and $A_k = V_k \times V_k$. For each edge $(i, j) \in A$

we assign a distance $d_{ij} \geq 0$ and a travel time $t_{ij} \geq 0$. Again, it is assumed that the travel times satisfy the *triangle inequality* i.e. $t_{ij} \leq t_{il} + t_{lj}$ for all $i, j, l \in V$. We assign a service time s_i and a time window $[a_i, b_i]$ to each node $i \in V$. The service time represents the time needed for loading and unloading and the time window indicates when the visit at the particular site must start; a visit to node i can only take place between time a_i and b_i. A vehicle is allowed to arrive to a site before the start of the time window but it has to wait until the start of the time window before the visit can be performed. For each node $i \in N$ we define l_i to be the amount of goods that should be loaded onto the vehicle at the particular node. We have that $l_i \geq 0$ for $i \in P$ and $l_i = -l_{i-n}$ for $i \in D$. Each vehicle $k \in K$ has a certain capacity C_k. Each node has assigned a *precedence number* Π_i. Nodes with low precedence must always be visited before nodes with higher precedence.

Each vehicle k should follow a legal route from its start terminal τ_k to its destination terminal τ'_k. A legal route $\overline{r}$ is a simple (loop-free) path

$$\overline{r} = (\tau_k = v_1, v_2, \ldots, v_h = \tau'_k) \tag{1}$$

satisfying the precedences and time windows at the customers, the capacity of the vehicle, and ensuring that a pickup takes place before a delivery, and that only requests serviceable by vehicle k are carried out.

More formally, we demand that a vehicle only visits nodes that can be serviced by the vehicle, i.e.

$$v_i \in N_k, \qquad i = 2, \ldots, h-1 \tag{2}$$

A pickup-delivery pair must be served by the same vehicle, and the pickup must take place before the delivery, hence we have

$$i \leq j, \qquad v_i \in P_k, v_j \in D_k, v_j = v_i + n \tag{3}$$

Precedences should be obeyed along the route, this is ensured by the constraints

$$i \leq j, \qquad \Pi_{v_i} \leq \Pi_{v_j} \tag{4}$$

To ensure that time windows are satisfied, we introduce $S_i \in \mathbb{R}_0^+$ to denote when the vehicle starts the service at site v_i. We then have the constraints

$$a_{v_i} \leq S_i \leq b_{v_i} \qquad i = 1, \ldots, h \tag{5}$$
$$S_{i+1} \geq S_i + s_i + t_{v_i, v_{i+1}} \qquad i = 1, \ldots, h-1 \tag{6}$$
$$a_{\tau_k} \leq S_1 \leq b_{\tau_k} \tag{7}$$
$$a_{\tau'_k} \leq S_h \leq b_{\tau'_k} \tag{8}$$

where $[a_{\tau_k}, b_{\tau_k}]$ is the time window of terminal τ_k and $[a_{\tau'_k}, b_{\tau'_k}]$ is the time window of terminal τ'_k. Finally, the capacity of the vehicle should be respected throughout the path. For this purpose we introduce $L_i \in \mathbb{R}_0^+$ to denote the load of the vehicle at node i after serving node i. Then we have

$$L_i \leq C_k \qquad i = 1, \ldots, h \tag{9}$$
$$L_{i+1} = L_i + l_{i+1} \qquad i = 1, \ldots, h-1 \tag{10}$$
$$L_1 = 0 \tag{11}$$
$$L_h = 0 \tag{12}$$

The *travel cost* of a given route $\overline{r}$ is

$$c_{\overline{r}} = \sum_{i=1}^{h-1} d_{v_i, v_{i+1}} \tag{13}$$

Situations may occur in which some requests cannot be serviced by the available vehicles. To model this situation we create n dummy routes, consisting of a single request. These routes do not make use of any vehicles but they have a large cost, denoted Γ. Requests that are not served by a vehicle are said to be located in the *request bank*.

The whole problem can now be formulated as follows: let R be the set of all feasible routes. The boolean matrix $(\alpha_{j\overline{r}})$ for $\overline{r} \in R$ and $j = 1, \ldots, n$ is used to indicate whether request j is serviced using route $\overline{r}$. The boolean matrix $(\beta_{k\overline{r}})$ for $\overline{r} \in R$ and $k = 1, \ldots, m$ is used to indicate whether the route $\overline{r}$ is carried out by vehicle k. Using binary variables $x_{\overline{r}}$ to indicate whether route $\overline{r}$ is used in the solution we get the following model

$$\min \quad f(x) = \sum_{\overline{r} \in R} c_{\overline{r}} x_{\overline{r}} \tag{14}$$

$$\text{s.t.} \quad \sum_{\overline{r} \in R} \alpha_{j\overline{r}} x_{\overline{r}} = 1 \qquad j = 1, \ldots, n \tag{15}$$

$$\sum_{\overline{r} \in R} \beta_{k\overline{r}} x_{\overline{r}} = 1 \qquad k = 1, \ldots, m \tag{16}$$

$$x_{\overline{r}} \in \{0, 1\} \qquad \overline{r} \in R \tag{17}$$

Note that a dummy route is not assigned to any vehicle, that is, for any dummy route $\overline{r}$ we have that $\beta_{k\overline{r}} = 0, \forall k = 1, \ldots, m$.

3 Problem transformations

The heuristic in this paper is applied to five different problems — VRPTW, CVRP, OVRP, MDVRP, SDVRP — which all are transformed to a RPDPTW. The conversions which will be described in the following paragraphs are extensions of the transformations presented by Ropke and Pisinger [50] for solving VRP problems with backhauls.

3.1 Vehicle Routing Problem with Time Windows

In order to transform a VRPTW instance to a RPDPTW instance we map every customer in the VRPTW to a request in the RPDPTW. Such a request consists of a pick up at the depot and a delivery at the customer site. The amount of goods that should be carried by the requests is equal to the demand of the corresponding customer. The time window of the pickup is set to $[a_d, a_d]$ where a_d is the start of the time window of the depot in the VRPTW and its service time is set to zero. The time window and service time of the delivery are copied from the corresponding customer in the VRPTW. In order to avoid routes that return to the depot for restocking we let all pickups and deliveries have precedence zero and one respectively. All vehicles in the RPDPTW have the same start and end terminals corresponding to the depot in the VRPTW. Distances and travel times in the RPDPTW are set in the natural way.

3.2 Capacitated Vehicle Routing Problem

A CVRP instance can easily be transformed to a VRPTW instance. This can for example be done by setting all travel and service times to zero and all time windows to [0,0]. If the CVRP contains a route duration constraint then travel times and durations should be set as in the CVRP. All time windows (including the ones at the end terminals) should be set to [0, D] where D is the route duration. The VRPTW is transformed to a RPDPTW as described in Section 3.1.

3.3 Site Dependent Vehicle Routing Problem

In the SDVRP a customer may only be serviced by a given subset of the vehicles, typically because the access paths to the node do not allow given vehicles to pass, or because specific facilities are demanded in the vehicle (e.g. a freezing compartment).

The SDVRP is easily modeled as a RPDPTW by using the transformation from CVRP to RPDPTW and noting that the RPDPTW allows us to specify the pickups P_k and deliveries D_k that can be carried out by vehicle k.

3.4 Open Vehicle Routing Problem

The OVRP is very close to the CVRP. The difference between the two problems is that in the OVRP the vehicles do not have to return to the depot. Thus an OVRP can be solved as an asymmetric CVRP by setting distances and travel times from every customer to the depot to zero.

The travel times in the resulting RPDPTW do not satisfy the triangle inequality, but our method is able to handle the problems anyway since $t_{ij} \leq t_{il} + t_{lj}$ is only violated when l is an end terminal. Our only reason for assuming that the triangle inequality is satisfied for the travel times is that we have to avoid situations in which the removal of one or more requests causes the travel time to increase. As the node sequence $i \rightarrow l \rightarrow j$ where $l \in \{\tau'_1, \ldots, \tau'_m\}$ never occurs in a valid route this violation of the triangle inequality does not cause any problems.

3.5 Multi-Depot Vehicle Routing Problem

In the MDVRP each customer may be serviced by a vehicle originating at any of the available depots. Even though our underlying RPDPTW model supports multiple depots, it requires that each request is assigned to a specific depot. In general this is a hard optimization problem of its own which needs to be handled together with the routing problem. Hence we use the following transformation:

Create a dummy base location where all routes start and end and where all ordinary requests are picked up. Also create a dummy request for each vehicle k in the problem. The pickup and delivery locations of these requests are located at the depot of the corresponding vehicle. A dummy request has demand zero, it does not have any service time and it can be served at any time. The set N_k of each vehicle k contains all ordinary requests and the dummy request corresponding to the vehicle. In this way we ensure that each vehicle will carry precisely one dummy request.

The precedences Π_i of a pickup and a delivery corresponding to an ordinary request are set to zero and two respectively. The precedence of the pickup and delivery of the dummy requests are set to one and three respectively. This ensures that all ordinary deliveries will be surrounded by the pickup and delivery of a dummy request. The distance and travel time between a pickup of an ordinary request and any other location is set to zero. All other distances and travel times are set as defined by the original MDVRP.

In a solution to the RPDPTW that serves all requests we know that each vehicle will begin at a start terminal located at the dummy base location, then perform a number of pickups and then go to the pickup of the dummy request. Next, the ordinary deliveries will be served and the vehicle will return to the delivery of the dummy request and then to the end terminal of the route. Before starting the pickup of the dummy request and after the delivery of it all travel times and distances will be zero. Furthermore travel times and distances are accumulated correctly while carrying the dummy request.

While solving MDVRP problems the cost of dummy routes Γ must be set to a sufficiently large number such that it will never be profitable to leave a dummy request in the request bank.

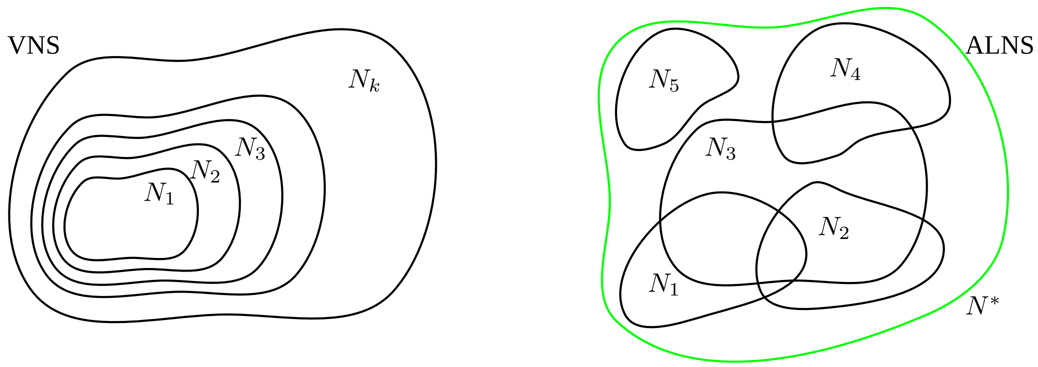


Figure 1: Illustration of neighborhoods used by VNS and ALNS. VNS typically operates on one type of neighborhood with variable depth while ALNS operates on structurally different neighborhoods $N_1, \ldots, N_k$ defined by the corresponding search heuristics. All neighborhoods $N_1, \ldots, N_k$ in ALNS are a subset of the neighborhood N^* defined by modifying q variables.

4 Adaptive Large Neighborhood Search

We will now describe the *Adaptive Large Neighborhood Search* (ALNS) framework used in the present paper. We believe that ALNS can be applied to a large class of difficult optimization problems, hence in the following we consider an optimization problem in the general IP form:

$$\min\{f(x) : Ax \leq b, x \in \mathbb{Z}^n\} \tag{18}$$

ALNS is a local search framework in which a number of simple algorithms compete to modify the current solution. In each iteration an algorithm is chosen to destroy the current solution, and an algorithm is chosen to repair the solution. The new solution is accepted if it satisfies some criteria defined by the local search framework applied at the master level.

To be more formal, we extend the domain of each variable x_i to $\mathbb{Z} \cup \{\perp\}$, where $\perp$ means undefined. A *destroy heuristic* chooses at most q variables which are assigned the value $\perp$. A repair heuristic assigns feasible values $x_i \in \mathbb{Z}$ to the q variables.

The ALNS framework is an extension of the *Large Neighborhood Search* presented by Shaw [53], where a large collection of variables are modified in each iteration. In ALNS the neighborhoods are searched by simple and fast heuristics. ALNS is also based on the *Ruin and Recreate* paradigm presented by Schrimpf et al. [52], or the *Ripup and Reroute* paradigm applied in [21]. In each iteration the current solution is partially destroyed and then repaired using some heuristics. ALNS also has similarities with *Very Large Neighborhood Search* (VLNS) presented by Ahuja et al. [1]. In VLNS the algorithm operates on very large neighborhoods chosen in a way so that they can still be searched efficiently.

Variable Neighborhood Search (VNS) was presented by Hansen and Mladenovic [31]. VNS makes use of a parameterized family of neighborhoods, typically obtained by using a given neighborhood with variable depth. When the algorithm reaches a local minimum using one of the neighborhoods, it proceeds with a larger neighborhood from the parameterized family. When the VNS algorithm gets out of the local minimum it proceeds with the smaller neighborhood. On the contrary, ALNS operates on a predefined set of large neighborhoods corresponding to the destroy (removal) and repair (insertion) heuristics. The neighborhoods are not necessarily well-defined in a formal mathematical sense — they are rather defined by the corresponding heuristic algorithm. The difference between VNS and ALNS is illustrated in Figure 1. In the sections that follow, we will distinguish between a neighborhood and the heuristic searching it.

Instead of viewing the ALNS heuristic as a sequence of destroy and repair operations one can alternatively see it as a sequence of *fix* and *optimize* operations. The fix operation selects a number of variables that are fixed at their current value; the optimize operation seeks to find a near-optimal solution that respects the fixed variables, that is, only non-fixed variables can be changed. After the optimization operation, all variables are unlocked again. The fix operation is analogous to the destroy operation and the optimize operation is analogous to the repair operation. The fix/optimize view might be helpful when applying the heuristic to problems where the destroy and repair operations do not seem intuitive.

4.1 Outline of algorithm

ALNS can be based on any local search framework, e.g. simulated annealing, tabu search or guided local search. The general framework is outlined in Figure 2, where lines 2–8 form the main loop of the local search framework at the master level. Implementing a simulated annealing algorithm is straightforward as one solution is sampled in each iteration of the ALNS. A simple tabu search could for example be implemented by randomly sampling a number of candidate solutions and choosing the best non tabu solution.

In each iteration of the main loop we choose one destroy and one repair neighborhood (line 3). An adaptive layer stochastically controls which neighborhoods to choose according to their past performance (score). The more a neighborhood N_i has contributed to the solution process, the larger score π_i it obtains, and hence it has a larger probability of being chosen.

The adaptive layer uses roulette wheel selection for choosing a destroy and a repair neighborhood. If the past score of a neighborhood i is denoted π_i and we have ω neighborhoods, then we choose neighborhood N_j with probability

$$\frac{\pi_j}{\sum_{i=1}^{\omega} \pi_i}$$

Adaptive Large Neighborhood Search

1 Construct a feasible solution x; set $x^* := x$
2 Repeat
3   Choose a destroy neighborhood N^- and a repair neighborhood N^+ using roulette wheel selection based on previously obtained scores $\{\pi_j\}$
4   Generate a new solution x' from x using the heuristics corresponding to the chosen destroy and repair neighborhoods
5   If x' can be accepted then set $x := x'$
6   Update scores π_j of N^- and N^+
7   If $f(x) < f(x^*)$ set $x^* := x$
8 Until stop criteria is met
9 Return x^*

Figure 2: Outline of the ALNS framework

Notice that the destroy and repair neighborhoods are selected independently, and hence two separate roulette wheel selections are performed.

In most applications the neighborhoods are searched by fast heuristics, hence it is reasonable to assume that they are equally fast. But if some heuristics are significantly slower than others, one may normalize the score π_i of a neighborhood with a measure of the time consumption t_i of the corresponding heuristic. This ensures a proper trade-off between time consumption and solution quality.

In line 4 of the ALNS-algorithm, we first destroy the current solution x using a heuristic searching the neighborhood N^- and then repair the solution using a heuristic corresponding to neighborhood N^+. It can be advantageous to use noising or randomization in the destroy and repair heuristics to obtain a proper diversification. In traditional local search heuristics the diversification is controlled implicitly by the local search paradigm (accept ratio, tabu list, etc.), but since we use large neighborhoods which are searched by simple heuristics, it is not sufficient to have a diversification operator at the master level. We also need a diversification operator at the sub-level to avoid stagnating search processes where the destroy and repair neighborhoods keep performing the same modifications to a solution.

Finally, in line 6 we update the scores π_i of the neighborhoods. A number of criteria can be used to measure how much a neighborhood contributes to the solution process: new best solutions are obviously given a large score, but also not previously visited solutions are given a score. Depending on the local search framework used on the master level, one may also give specific scores to accepted solutions e.g. in a simulated annealing framework. Since each step of the ALNS heuristic involves two neighborhoods (a destroy and a repair neighborhood), the score obtained in a given iteration is divided equally between them.

Every M iterations of the ALNS algorithm, the scores π_i are reset, and the probabilities for choosing the neighborhoods are recalculated. Each neighborhood is assigned a minimum probability for being chosen to ensure that statistical information about its performance can be collected. The probabilities for choosing a neighborhood can also be a weighted sum of the score during the last M iterations, and the overall score since the beginning of the algorithm.

4.2 Designing an ALNS algorithm

In order to design an ALNS algorithm for a given optimization problem one needs to

- Choose a number of fast construction heuristics which are able to construct a full solution given a partial solution (a solution where some variables are set to $\perp$ and some have a real value).
- Choose a number of destroy heuristics. It might be worthwhile to choose destroy heuristics that are expected to work well with the chosen construction heuristics, but it is not necessary.
- Choose a local search framework at the master level.

In each iteration the heuristic corresponding to a *destroy neighborhood* should remove a given number q of variables. The destroy neighborhoods (N^-) should be a proper mix of neighborhoods which can intensify and diversify the search. To diversify the search, one may randomly choose q decision variables, i.e. using a *random removal* neighborhood. To intensify the search one may try to remove q "critical" variables, i.e. variables having a large cost or variables spoiling the current structure of the solution (e.g. edges crossing each other in a Euclidean traveling salesman problem). This is known as *worst removal* or *critical removal*. Concrete examples on *random removal* and *worst removal* neighborhoods in a VRP context are given in Sections 5.1.1–5.1.2.

One may also choose a number of related variables that are easy to interchange while maintaining feasibility of the solution. This *related removal* neighborhood was introduced by Shaw [53]. More formally we can measure the relatedness r_{ij} of two variables x_i and x_j by the deviation of the corresponding coefficients in the constraint matrix A in problem (18). The smaller r_{ij} the more related are variables x_i and x_j. How exactly r_{ij} should be defined depends on the concrete problem at hand, and one may even have several simultaneous neighborhoods defined by various choices of the relatedness measure (r_{ij}). In order to choose the q most related variables, one needs to solve the NP-hard *dispersion-sum problem* given by

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} r_{ij}x_i x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} x_j = q \\ & x_j \in \{0,1\}, \quad j = 1,\dots,n \end{aligned} \tag{19}$$

A greedy heuristic for this problem running in $O(n^3)$ was presented in [44] together with a more time-consuming exact algorithm. If n is large, it may be too time-consuming even to compute the whole matrix (r_{ij}) and one will instead choose related variables according to some heuristics. Shaw [53] presented an algorithm running in $O(qn)$ time by initially selecting a variable at random, and then repeatedly selecting an already selected variable i and finding a variable j which minimizes r_{ij} and adding j to the set of chosen variables. An alternative heuristic is based on a modified Kruskal's algorithm for the minimum spanning tree problem, using r_{ij} as edge weights, which stops when a connected component with q or more elements has been constructed. The variables in this component are set to $\perp$. The worst-case running time of this algorithm is $O(n^2 \log n)$ as we have n^2 edges in Kruskal's algorithm. Ropke and Pisinger [49] used a modified version of this algorithm in the VRP for splitting requests on a route into two strongly connected subsets. It should be noted that solving the dispersion sum problem (19) to optimality seldom would be a good idea even if it could be done in a very short time. If r_{ij} is independent of the current solution the destroy neighborhood obtained by solving the dispersion sum problem to optimality would always assign $\perp$ to the same set of variables. Concrete examples on various *related removal* neighborhoods are given in Sections 5.1.3–5.1.5.

Following the same idea as in *related removal* one may choose a number of variables having small coefficients in the resource constraints in (18), as these generally are easy to interchange and loosely speaking can fill up unused resource constraints. We denote this strategy *small removal*.

Finally, one may use *history based removal* where the q variables are chosen according to some historical information as presented in [49]. The historical information could for example count how often setting a given variable (or set of variables) to a specific value leads to a bad solution. One may then try to remove variables that currently are assigned an improper value, based on the historical information. Variants of the *history based removal* neighborhood are discussed in Sections 5.1.6–5.1.7.

repair neighborhoods (N^+) are typically based on concrete well-performing heuristics for the given problem. These heuristics can make use of variants of the greedy paradigm, e.g. performing the locally best choice in each step, or performing the least bad choice in each step. An alternative variant of the greedy paradigm is to set all variables to their upper bound in problem (18), and repeatedly decrease the most expensive variable until a feasible solution is obtained. The repair heuristics can also be based on approximation algorithms or exact algorithms which have been relaxed to obtain faster solution times at the cost of solution quality. Shaw [53] and Bent and Van Hentenryck [2] proposed more expensive algorithms like searching N^+ based on relaxed branch-and-bound methods. Although ALNS mainly is intended to use cheap heuristics, more expensive search methods can be used if the scores of the corresponding neighborhoods are normalized with respect to the time consumption. In the context of VRP problems, repair neighborhoods are considered in more detail in Section 5.2 discussing both simple greedy approaches and variants of regret heuristics.

Some optimization problems can be split into a number of sub-problems, where each sub-problem can be solved individually. Such problems include the Bin Packing Problem in which a number of bins are to be filled, or the Vehicle Routing Problem in which a number of routes are to be constructed. For such problems one should decide whether the subproblems should be solved one by one (*sequential heuristics*) or all subproblems should be solved at the same time (*parallel heuristics*). Sequential heuristics are easier to implement but may have the disadvantage that the last subproblem solved is left with variables that do not fit well together. This is to some extent avoided in parallel heuristics.

A natural extension to the ALNS framework is to have *coupled neighborhoods*. In principle one may, for each destroy neighborhood N_i^-, define a subset $K_i \subseteq \{N^+\}$ of repair neighborhoods that can be used with N_i^-. The roulette wheel selection of repair neighborhoods will then only choose a neighborhood in K_i if N_i^- was chosen.

As a special case, one may have $K_i = \emptyset$ meaning that the neighborhood N_i^- takes care of both the destroy and repair steps. One could use an ordinary local search heuristic to compete with the other destroy and repair neighborhoods, ensuring that a thorough investigation of the solution space close to the current solution is made from time to time.

For some problems it may be sufficient to have a number of destroy and repair heuristics that are selected randomly with equal probability, that is without the adaptive layer. We will denote such a heuristic a *Large Multiple-Neighborhood Search* (LMNS). The LMNS heuristics share the robustness of the ALNS heuristics, while having considerably fewer parameters to calibrate.

4.3 Properties of the ALNS framework

The ALNS framework has several advantages. For most optimization problems we already know a number of well-performing heuristics which can form the core of an ALNS algorithm. Due to the large neighborhoods and diversity of the neighborhoods, the ALNS algorithm will explore large parts of the solution space in a structured way. The resulting algorithm becomes very robust, as it is able to adapt to various characteristics of the individual instances, and seldom is trapped in a local minima.

ALNS is particularly well suited for tightly constrained problems, in which small neighborhoods are not sufficient to escape a local minima or certain areas of the solution space. In such problems, the large neighborhood search makes it possible to change many variables each time to reach new feasible solutions.

The calibration of the ALNS algorithm is quite limited as the adaptive layer automatically adjusts the influence of each neighborhood used. It is still necessary to calibrate the individual sub-heuristics used for searching the destroy and repair neighborhoods, but one may calibrate these individually or even use the parameters used in existing algorithms.

In the design of most local search algorithms the researcher has to choose between a number of possible neighborhoods. In ALNS the question is not “either-or” but rather “both-and”. As a matter of fact, our experience is that the more (reasonable) neighborhoods the ALNS heuristic makes use of, the better it performs [49].

5 ALNS applied to the RPDPTW

We will now describe how the general ALNS framework has been adapted to the RPDPTW problem. The “variables” in the ALNS framework correspond to requests in the RPDPTW. A destroy neighborhood N^- consists of removing q requests from the existing routes and assigning them to the *request bank*.

The heuristic described in this section is almost identical to the heuristic used to solve a large class of vehicle routing problems with backhauls ([50]). One more destroy heuristic has been added (see section 5.1.5) and the formula determining the number of requests to remove has been changed (see section 6.1.1).

For completeness we will describe the various heuristics associated with the destroy neighborhoods in Section 5.1. A repair neighborhood N^+ inserts requests from the request bank into one or more legal routes. The associated insertion heuristics are described in Section 5.2. The local search framework used at the master level is simulated annealing to be described in Section 5.3. Section 5.4 describes the noising method used to diversify the search of the heuristics. Finally, the scheme used for adjusting the weights in the roulette wheel selection is described in Section 5.5.

5.1 Request removal

The ALNS heuristic for the RPDPTW makes use of seven different removal heuristics, each searching a given removal neighborhood N^-. The heuristics take as input a given solution x and outputs q requests that have been removed from the routes.

5.1.1 Random removal

The simplest removal heuristic, `random removal`, selects q requests at random and removes them from the solution. This obviously has the effect of diversifying the search.

5.1.2 Worst removal

The purpose of the `worst removal` heuristic is to choose a number of requests that are very expensive, or which somehow spoil the structure of the current solution. In the RPDPTW it seems reasonable to try to remove requests with high cost and insert them at another place in the solution to obtain a better solution value.

Given a request i served by some vehicle in a solution x we define the cost of the request Δf_{-i} as the difference between the value of $f(x)$ and the cost of solution x where request i is removed completely from the problem.

The `worst removal` heuristic now repeatedly chooses a new request i, having the largest cost Δf_{-i} until q requests have been removed. The removal heuristic is randomized, the randomization is controlled by the parameter p. If p is small, the most expensive request is selected, while less expensive requests may be chosen for larger values of p with a probability that decreases with the cost Δf_{-i}. We refer to [49] for additional details.

5.1.3 Related removal

The purpose of the `related removal` heuristic is to remove a set of requests that in some sense are *related* and hence easy to interchange. For the RPDPTW we define the relatedness r_{ij} of two orders i and j solely by the distance between the requests, as introduced by Ropke and Pisinger [49]. Since each request i consists of a pickup node i and a delivery node $i+n$ we get the expression

$$r_{ij} = \frac{1}{D}\left(d'(i,j) + d'(i,j+n) + d'(i+n,j) + d'(i+n,j+n)\right) \tag{20}$$

where the distance measure $d'(u,v)$ between two nodes in this context is defined as

$$d'(u,v) = \begin{cases} d_{uv} & \text{if } u \text{ and } v \text{ are not located at a terminal} \\ 0 & \text{if } u \text{ or } v \text{ is located at a terminal} \end{cases} \tag{21}$$

The motivation for neglecting the distance from a terminal is that the terminal is going to be visited in any case, and hence should not contribute to the relatedness measure of two requests.

The denominator D is set to the number of nonzero terms in equation (20), i.e. the number of pickups and deliveries taking place at a site different from a terminal. Hence if all nodes are different from a terminal we set $D := 4$ while if both requests have a pickup at a terminal we set $D := 1$.

The relatedness measure is used to remove customers as described in Shaw [53]. The algorithm initially selects a request i by random. Then it repeatedly chooses an already selected request j and selects a new request which is most related to j. The algorithm stops when q requests have been chosen. Like in the worst removal heuristic (Section 5.1.2) the process is controlled by a randomization parameter p. If p is zero, the most related request is always chosen in the inner loop. If $p > 0$ a less related request may be chosen, where the probability of choosing a request decreases with the relatedness measure r_{ij} and increases with p. The algorithm is described in more detail in [49].

5.1.4 Cluster removal

The cluster removal heuristic is a variant of the related removal heuristic in which we try to remove clusters of related requests from a few routes. As a motivation, consider a route where the requests are grouped into two geographical clusters. When removing requests from such a route it is often important to remove one of these clusters entirely as the insertion methods otherwise would be prone to insert the removed requests back into the route. The related removal heuristic from Section (5.1.3) has a tendency to leave requests from such a cluster on the original route so therefore we propose a heuristic that seeks to remove an entire cluster at once.

Although we could use the same algorithm as above for selecting related requests — just restricted to a single route — we have chosen to use a heuristic based on strongly connected components, as described in Section 4.2. We simply run Kruskal's algorithm for the minimum spanning tree problem (using r_{ij} for the edge distances) and terminate the algorithm when two connected components remain. One of these clusters is chosen at random and the requests from the chosen cluster are removed. If less than q requests have been selected, we randomly pick a removed request and choose a request from a different route, that is most related to the given request. The route of the new request is partitioned into two clusters and so the process continues until the desired number of requests has been removed. We refer the reader to Ropke and Pisinger [50] for more details.

5.1.5 Time-oriented removal

The time oriented removal is another variant of the related removal heuristic. In this heuristic we try to remove requests that are served at roughly the same time as we hope that these requests are easy to interchange.

The heuristic works as follows. A request $\tilde{r}$ is chosen at random and the B requests that are closest to $\tilde{r}$ (according to the distance r_{ij} defined in (20)) are marked. We define a time-oriented distance between two requests as

$$\Delta t_{ij} = |t_{p_i} - t_{p_j}| + |t_{d_i} - t_{d_j}| \tag{22}$$

where t_{p_i} and t_{d_i} are the times of the pickup and the delivery of request i in the current solution. Among the B marked requests we select the $q - 1$ that are closest to $\tilde{r}$ according to Δt_{ij}. The process is controlled by a randomization parameter p like in the related removal heuristic described in Section 5.1.3. These requests are removed together with $\tilde{r}$.

Before running the removal heuristic we first select a subset of all requests that are geographically close to the chosen request, as we observed that this selection made the heuristic perform better on large instances. The reason for this is that if the heuristic only considered requests that are close to the chosen request time-wise, then only one or two requests would be removed from each route in the larger problems, and this makes it hard to make any major improvements to the solution.

5.1.6 Historical node-pair removal

It is well-known from several metaheuristics that using historical information in the local search (e.g. the long term memory or the aspiration level in tabu search) may improve the performance of a local search algorithm. In the present heuristic we look at the historical success of visiting two nodes right after each other in a route, while the heuristic in Section 5.1.7 looks at the historical success of servicing two requests by the same vehicle.

The historical node-pair removal heuristic (denoted the *neighbor graph removal heuristic* in [50]) makes use of both historical information and the present solution when removing the requests. With each pair of nodes $(u, v) \in A$ we associate a weight $f^*_{(u,v)}$ which indicates the best solution value found so far, in a solution which used edge (u, v). Initially $f^*_{(u,v)}$ is set to infinity, and each time a new solution is found, we update the weights $f^*_{(u,v)}$ of all edges used in the given solution, for which the edge weight can be improved.

We may use the edge weights $f^*_{(u,v)}$ to remove requests that seem to be misplaced. The removal heuristic simply calculates the cost of a request $(i, i+n)$ in the current solution by summing the weights of edges incident to i and $i+n$. The most costly request is removed, and the process is repeated until q requests have been extracted. To ensure some variation in the extracted requests, randomness is introduced in the removal process.

5.1.7 Historical request-pair removal

An alternative history-based removal heuristic can make use of the historical success of placing pairs of requests in the same route. We will call this approach `historical request-pair removal` (denoted *request graph removal* in [50]).

For this purpose we introduce the weight $h_{(a,b)}$ for each pair of requests $(a, b) \in \{1, \ldots, n\} \times \{1, \ldots, n\}$. The weight $h_{(a,b)}$ denotes the number of times the two requests a and b have been served by the same vehicle in the B best unique solutions observed so far in the search. Initially $h_{(a,b)}$ is set to zero, and each time a new unique top-B solution is observed, the weights are incremented and decremented according to the solutions entering and leaving the top-B solutions. An appropriate value for B was experimentally found to be 100.

The weights $h_{(a,b)}$ could be used in a similar way as in the historical node-pair removal heuristic described above, but initial experiments indicated that this was an unpromising approach. Instead, the graph is used to define the relatedness between two requests, such that two requests are considered to be related if the weight of the corresponding edge in the request graph is high. This relatedness measure is used as in the related removal heuristic described in Section 5.1.3.

5.2 Inserting requests

The considered insertion heuristics all construct a number of routes for the vehicles. As each route can be considered as an individual sub-problem the heuristics can build the routes *sequentially* or *in parallel* as discussed in Section 4.2. The sequential heuristics build one route at a time while parallel heuristics construct several routes at the same time. The heuristics presented in this paper are all parallel, as they are used in a context where a number of partial routes $k \in R$ are given, and a number of unplaced requests U is inserted from the request bank.

5.2.1 Basic greedy heuristic

A simple greedy approach is to repeatedly insert a request in the cheapest possible route. More formally, let $\Delta f_{i,k}$ denote the change in the objective value incurred by inserting request i at the *cheapest* position in route k. We set $\Delta f_{i,k} = \infty$ if request i cannot be inserted in route k. Following the greedy approach we calculate

$$(i, k) := \arg \min_{i \in U, k \in R} \Delta f_{i,k} \tag{23}$$

and insert request i in route k at its minimum cost position. This process continues until all requests have been inserted or no more requests are feasible. The time complexity of this `basic greedy` heuristic is decreased by tabulating all values of $\Delta f_{i,k}$ and noting that only one route is changed in each iteration.

5.2.2 Regret heuristics

An obvious problem with the `basic greedy` heuristic is that it often postpones the placement of difficult requests to the last iterations where we do not have much freedom of action. The `regret` heuristic tries to circumvent the problem by incorporating a kind of look-ahead information when selecting the request to insert. Regret heuristics have been used by Potvin and Rousseau [45] for the VRPTW and in the context of the Generalized Assignment Problem by Trick [59].

Let Δf_i^q denote the change in the objective value incurred by inserting request i into its best position in the qth *cheapest* route for request i. For example Δf_i^2 denotes the change in the objective value by inserting request i in the route where the request can be inserted *second cheapest*. In each iteration, the regret heuristic chooses to insert the request i according to:

$$i := \arg\max_{i \in U} \left(\Delta f_i^2 - \Delta f_i^1 \right) \tag{24}$$

The request is inserted in the best possible route at the minimum cost position. In other words, we maximize the difference of cost of inserting the request i in its best route and its second best route. We repeat the process until no more requests can be inserted.

The heuristic can be extended in a natural way to define a class of regret heuristics: the `regret-`q heuristic is the construction heuristic that in each construction step chooses to insert request i given by:

$$i := \arg\max_{i \in U} \left(\sum_{h=2}^{q} \Delta f_i^h - \Delta f_i^1 \right) \tag{25}$$

Ties are broken by selecting the request with smallest insertion cost. The request i is inserted at its minimum cost position, in its best route.

The `regret` heuristic based on criteria (24) is obviously a `regret`-2 heuristic and the `basic greedy` heuristic from Section 5.2.1 is a `regret`-1 heuristic due to the tie-breaking rules. Informally speaking, heuristics with $q > 2$ investigate the cost of inserting a request on the q best routes and chooses to insert the request whose cost difference between inserting it into the best route and the $q - 1$ best routes is largest. Compared to a `regret`-2 heuristic, `regret-`q heuristics with large values of q discover earlier when the possibilities for inserting a request at a favorable place becomes limited.

5.3 Master local search framework

At the master level we have chosen to use simulated annealing as our local search framework. Our acceptance criteria in Line 5 of the main algorithm depicted in Figure 2 thus becomes to accept a candidate solution x' given the current solution x with probability

$$e^{-\frac{f(x')-f(x)}{T}}, \tag{26}$$

where $T > 0$ is the *temperature*. We use a standard exponential cooling rate, starting from the temperature T_{start} and decreasing T according to the expression $T = T \cdot c$, where c is the *cooling rate*, $0 < c < 1$. We calculate T_{start} by inspecting our initial solution. The following method was developed in [50] and works well when the number of requests in the problems to be solved is relatively constant. First the cost z' of the initial solution is calculated using a modified objective function. In the modified objective function, Γ (cost of having requests in the request bank) is set to zero. The start temperature is now set such that a solution that is w percent worse than the current solution is accepted with probability 0.5. The reason for setting Γ to zero is that typically this parameter is large and could cause us to set the starting temperature too high if the initial solution had some requests in the request bank. Now w is a parameter that has to be set. We denote this parameter the *start temperature control parameter*. We have observed that this approach is better at coping with instances of different sizes if we divide the start temperature found by the number of requests in the instance.

5.4 Applying noise to the objective function

As mentioned in Section 4.1 it can be necessary to use noising or randomization in the destroy and repair heuristics, as a diversification operator at the master level is not sufficient.

For the RPDPTW problem we have chosen to add a noise term to the objective function of the insertion heuristics. Every time we calculate the cost C of a request insertion into a route, we add some noise δ and calculate a modified insertion cost $C' = max\{0, C + \delta\}$. The noise δ is chosen as a random number in the interval $[-N_{\max}, N_{\max}]$, where $N_{\max} = \eta \cdot \max_{i,j \in V}\{d_{ij}\}$, and η is a parameter that controls the amount of noise. We use the maximum distance to make the noise level proportional to the objective value. The distances form part of the objective function in all problems considered, hence the noise level is somehow proportional to the objective function.

Every insertion heuristic is split into two heuristics — one using noise, and one using the original objective function only. After selecting which removal and insertion heuristic to use, it is decided if the clean or the noise imposed insertion heuristic should be used. This is again done using the roulette wheel selection principle as we keep track of how well the insertion heuristics with and without noise have been performing recently. Notice that we do not keep track of how well each individual insertion heuristic is performing with and without noise, but only the insertion heuristics in general.

5.5 Adaptive weights adjustment

The roulette wheel selection mechanism in the ALNS framework presented in Section 4.1 is based on the scores π_i of the respective heuristics. A high score corresponds to a successful heuristic, and hence the heuristic should be chosen with larger probability.

The scores are collected during some small time segments, defined as 100 iterations. The *observed* score $\overline{\pi}_{i,j}$ of a heuristic i in time segment j is incremented with the following values depending on the new solution x':

- σ_1 The last remove-insert operation resulted in a new global best solution x'.
- σ_2 The last remove-insert operation resulted in a solution x' that has not been accepted before, and the cost of the new solution is better than the cost of current solution.
- σ_3 The last remove-insert operation resulted in a solution x' that has not been accepted before. The cost of the new solution is worse than the cost of current solution, but the solution was accepted.

We distinguish between the two latter situations since we prefer heuristics that are able to improve on the solution, but we also want to reward heuristics that can diversify the search to some extent. We keep track of visited solutions by assigning a hash key to each solution and storing the key in a hash table.

At the end of each segment we calculate the *smoothened* scores to be used in the roulette wheel selection as

$$\pi_{i,j+1} = \rho\frac{\overline{\pi}_{i,j}}{a_i} + (1-\rho)\pi_{i,j} \tag{27}$$

where a_i is the number of times the heuristic has been called in the time segment. The *reaction factor* ρ controls how quickly the weight adjustment algorithm reacts to changes in the scores. If $\rho = 1$ then the roulette wheel selection is only based on the scores in the most recent segment, while if $\rho < 1$ the scores of past segments is also taken into account. For an illustration of how the scores evolve during a search we refer the reader to [49].

5.6 Minimizing the number of vehicles used

The presented heuristic minimizes the travel costs, hence in order to minimize the number of vehicles also, we use a two-stage approach.

Starting from a heuristic solution which makes use of m vehicles, we repeatedly remove one route and place the corresponding requests in the request bank. If the ALNS heuristic is able to find a solution that serves all requests we proceed with a lower number of routes. We assign a large cost Γ to requests in the request bank to encourage solutions with all requests serviced.

If the ALNS heuristic fails to find a solution with all requests serviced, the algorithm steps back to the last feasible solution encountered and proceeds with the second stage of the algorithm which consists of the ordinary ALNS heuristic with the last found feasible solution as a starting point. For additional detail on the two-stage algorithm see [49].

A different two-stage approach was used by Bent and Van Hentenryck [2], in which two distinct neighborhoods and metaheuristics were used for the two stages.

5.7 Initial solution

The initial solution used in the local search is found by a regret-2 heuristic. All requests are initially placed in the request bank, and the regret-2 heuristic is run in parallel for all vehicles.

6 Computational experiments

6.1 Parameter tuning

In order to keep the parameter tuning to a minimum we have used almost the same parameter setting as determined in [49], with the exception of the cooling rate c and the start temperature control parameter w. These were calibrated by selecting 5 reasonable values for each parameter and testing the 25 possible combinations on 8 VRPTW instances with between 100 and 1000 customers. This was done separately for both the vehicle minimizing ALNS and the ordinary distance minimizing ALNS, so different values for c and w are used when trying to find a feasible solution and when minimizing the distance.

6.1.1 Selecting the number of requests to remove

In our past work [49, 50] we have removed up to 100 requests in each iteration. Experiments indicated that we seldom accepted the moves resulting from such removals as the insertion heuristics are too weak. Consequently the maximum number of requests that can be removed in a single iteration has been reduced to 60. It was also observed that moves resulting from removing a small number of requests often were accepted, but seldom lead to any major improvements of the solution. Therefore we now remove at least $0.1n$ requests in each iteration. To be precise, the number of requests to remove is found as a random number between $\min\{0.1n, 30\}$ and $\min\{0.4n, 60\}$. That is, for small instances the number of requests to remove will be in the interval $[0.1n, 0.4n]$ while for larger instances the interval is $[30, 60]$.

6.2 Analysis of typical search

In order to illustrate how the present ALNS heuristic works, we have produced a number of figures by running the heuristic on a 200 customer VRPTW instance minimizing the traveled distance. All figures are from the same search.

Figure 3 shows the cost of the accepted solutions and the best known solution as a function of the iteration count. The figure is very typical for a Simulated Annealing metaheuristic. Initially very poor moves are accepted and consequently the graph of accepted solutions is fluctuating wildly. As the temperature is decreased the fluctuations become smaller and they eventually nearly die out such that only improving solutions or very mildly deteriorating solutions are accepted.

The next sequence of figures all show the distance between selected solutions. We have chosen to define the distance between two solutions x and x' as the Hamming distance between the corresponding binary edge-variables. Figure 4 (left) shows the distance between each new accepted solution and the previously accepted solution (the current solution). The figure illustrates that in the first half of the search the ALNS can make huge changes to the solution in a single move as discussed in Section 4.3. In the other half of the search only small moves are accepted. Figure 4 (right) depicts the difference between each proposed solution and the last accepted solution. The figure shows that large moves are proposed throughout the search process, but toward the end of the search these large moves are not accepted.

The above observations cause us to suggest some possible improvements to the algorithm: (1) To wards the end of the search it seems to be beneficial to reduce the number of requests q that are removed in each iteration as the simulated annealing framework generally only will accept minor changes. This could speed up the algorithm or allow us to perform more iterations within the same amount of time. (2) Several moves have distance zero, meaning that no changes were made to the solution vector. Obviously, such moves should be avoided, possibly by incorporating a tabu-like principle in the insertion heuristics.

Figure 5 (top left) shows the Hamming distance between the accepted solutions and the previously best known solution. Every time the distance reaches zero we have most likely found a new best solution (or we have returned to the previously best known solution). It is interesting to see how quickly the search

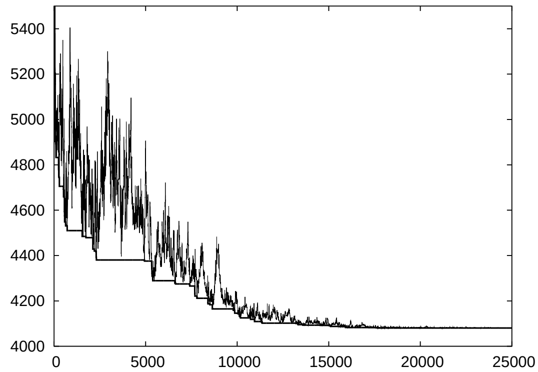


Figure 3: Solution cost as function of iteration count. Along the x-axis we show the iteration count while the y-axis shows solution cost. The upper graph is the cost of the accepted solutions while the lower graph is the cost of the currently best known solution.

moves away from the currently best known solution. This behavior is contrary to some of the ideas behind the Variable Neighborhood metaheuristics and the Noising Method, where one tries to stick around the currently best known solution or return to it if the current search direction seems fruitless. Also notice that we move very far away from the best solutions. This can be seen as the number of edges in a solution is equal to $2n + m$. The maximum Hamming distance between two solutions is therefore $2(2n + m)$. In the instance studied in this section $n = 200$ and $m = 20$, thus the maximum hamming distance for this instance is 840.

Figure 5 (top right) shows the Hamming distance from each accepted solution and the best solution found throughout the search. It is interesting to see that this plot is much more steady compared to the plot in Figure 5 (top left) and that even though we are moving very far away from the previously best known solution, the distance to the overall best solution (which of course is unknown early in the search) remains roughly stable.

Figure 5 (bottom) combines the two previous plots. The upper contours of the two plots fit each other surprisingly well. This indicates that the ALNS heuristic quickly moves away from the currently best known solution until the distance to the currently best known solution is roughly the same as the distance to the final best known solution. The search then visits solutions where the two distances are roughly the same until a new best solution is found. We believe that the Simulated Annealing framework is responsible for this behavior.

6.3 Application of the heuristic to standard benchmark problems

In this section we examine how the proposed heuristic performs on standard benchmark instances for the five problem types considered in this paper. In order to investigate how much influence the number of LNS iterations has on the solution quality, we have tested two configurations of our algorithm. One version (*ALNS-25K*) that does 25000 iterations while minimizing the total traveled distance and one that does 50000 iterations (*ALNS-50K*). Both configurations use up to 25000 iterations in the vehicle minimization stage. The cooling rate c in the simulated annealing algorithm described in Section 5.3 was adjusted such that both configurations go through the same temperature span.

We have applied the heuristic to each instance 5 or 10 times, depending on the instance size. We report the best solution value out of the 5 or 10 experiments as well as the average solution value.

All experiments were performed on a 3GHz Pentium 4 computer. Detailed results from the experiments can be found in the appendix. As mentioned before, the same parameter configuration has been used for all experiments.

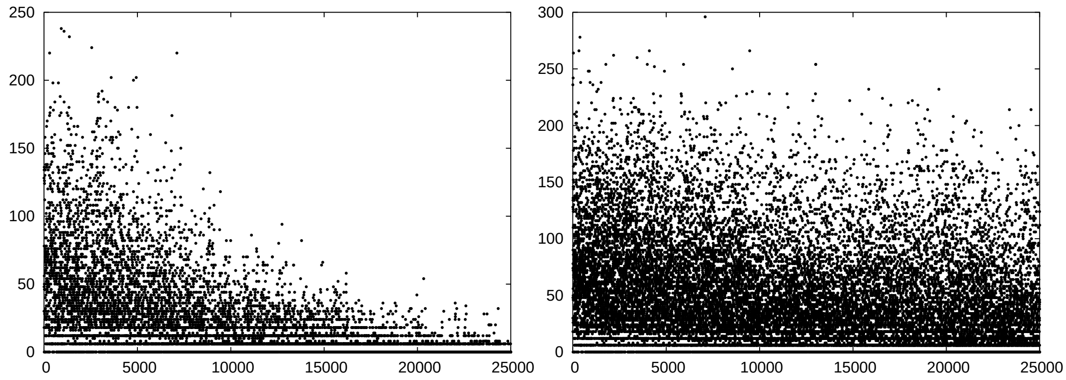


Figure 4: **Left:** Difference between accepted solutions. The figure shows the Hamming distance between an accepted solution and the last accepted solution. **Right:** Difference between proposed solution and last accepted solution. The figure shows the Hamming distance between each proposed solution and the last accepted solution. The x-axis shows iteration count and the y-axis shows solution distance.

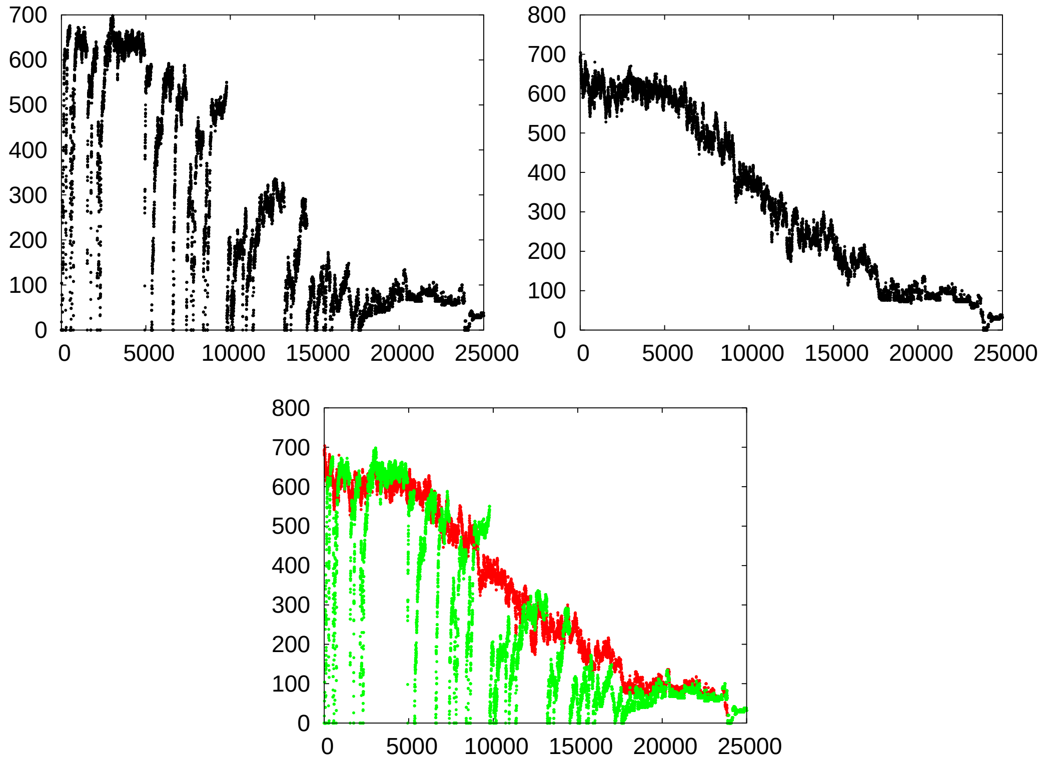


Figure 5: **Top left:** Hamming distance between accepted solutions and the currently best known solution. **Top right:** Hamming distance between accepted solutions and the best solution found during the search. **Bottom:** The two plots showed in the same diagram. The x-axis shows iteration count and the y-axis shows Hamming distance.

6.3.1 Vehicle routing problems with time windows (VRPTW)

A large number of metaheuristics have been proposed for solving the VRPTW. Bräysy and Gendreau [8] have surveyed most of these approaches, and their survey contains 47 metaheuristics. Most of these metaheuristics have been applied to the *Solomon data set* [55]. The Solomon data set contains 56 VRPTW instances that all contain 100 customers. The instances contain a variety of customer and time window distributions and have proved to be a challenge for both heuristics and exact methods since their introduction. Most of the proposed metaheuristics use vehicle minimization as primary objective and travel distance minimization as secondary objective, we prioritize our objectives in the same way. In this section we compare the ALNS heuristic to the “best” of the previously proposed metaheuristics. It is hard to decide which of the previously proposed metaheuristics that are the best, as several criteria for comparing the heuristics could be used. In this paper we have selected the metaheuristics that have been able to reach the minimum number of total vehicles used for all of the instances in the Solomon data set, as these in a certain sense can be regarded as the best heuristics in terms of solution quality. Table 1 summarizes this comparison.

The table shows that the ALNS heuristic is able to compete with the best heuristics for the VRPTW when considering the moderately sized Solomon instances, even though it was not specifically designed for this problem type. The heuristics by Homberger and Gehring [32] and Bent and Van Hentenryck [2] obtain slightly better results compared to the best solutions obtained by ALNS-25K, but the papers do not state how many experiments that were performed to reach these results. On the other hand, ALNS-25K reaches slightly better solutions than the three remaining heuristics and the computational time is reasonable. The column showing the average performance of ALNS-25K indicates that a single run of the heuristic can be performed quite fast but then one should not expect to reach the minimum number of vehicles. It does not seem worthwhile to spend 50000 iteration instead of 25000 for these rather small problems. During the calibration of the algorithm we discovered a new best solution to problem R207. This solution can be found in the Appendix.

When the VRPTW has been solved by exact methods in the literature one has usually considered minimizing the traveled distance without putting any limits on the number of vehicles. Furthermore all distances are usually truncated to one decimal (see for example the work by Larsen [39]). In Table 2 we summarize the result of applying the ALNS-25K heuristic to the Solomon VRPTW instances using the same objective and rounding criteria as the exact methods. The heuristic has been applied to each instance 10 times and the table reports the best and average performances. The table shows that the heuristic is able to find solutions that are very close to the optimal solutions and in many cases the heuristic is able to identify the optimal solution in at least one of the test runs.

The optimal solutions have been collected from Chabrier [10], Cook and Rich [14], Danna and Le Pape [20], Feilet et al. [24], Irnich and Villeneuve [34], Kallehauge et al. [35], Kohl et al. [36] and Larsen [39].

Larger VRPTW instances have been proposed by Gehring and Homberger [27]. The Gehring/Homberger data set contains 300 instances with between 200 and 1000 customers. In Tables 3–7 we compare the ALNS heuristic to the best heuristics that have been applied to these problems. The two heuristics that reach the best solution quality is the heuristic by Mester and Bräysy [42] and the ALNS heuristic. Overall the ALNS heuristic is better at minimizing the number of vehicles which is the primary objective of these problems. The heuristic of Mester and Bräysy is very good at minimizing the traveled distance though. The experiments show that the time used by the ALNS heuristic scales quite well with the problem size when the number of iterations is kept fixed. The 50000 iteration ALNS configuration becomes worthwhile for the larger problems. For problems with 600 customers or more the difference in total traveled distance obtained by the ALNS-25K and ALNS-50K configurations become quite large, as the simulated annealing metaheuristic needs more iterations to obtain a good solution for large problems.

The ALNS heuristic has been able to improve the best known solution for 122 out of the 300 large scale VRPTW instances. The best solutions for the large VRPTW instances obtained by the ALNS-25K and ALNS-50K configurations are shown in Table 8.

6.3.2 Multi depot vehicle routing problem (MDVRP)

Table 9 shows the results obtained on 33 MDVRP instances used by Cordeau et al. [17]. Both ALNS configurations have been applied 10 times to each instance. The results obtained by the ALNS heuristic

	BBB	HG	B	BH	IIKMUY	ALNS 25K		ALNS 50K	
R1	11.92	11.92	11.92	11.92	11.92	11.92	12.03	11.92	12.03
	1221.10	1212.73	1222.12	**1211.10**	1217.40	1213.39	1216.93	1212.39	1215.16
R2	2.73	2.73	2.73	2.73	2.73	2.73	2.75	2.73	2.75
	975.43	955.03	975.12	**954.27**	959.11	958.60	968.01	957.72	965.94
C1	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	828.48	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**
C2	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00
	589.93	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**
RC1	11.50	11.50	11.50	11.50	11.50	11.50	11.60	11.50	11.60
	1389.89	1386.44	1389.58	**1384.17**	1391.03	1385.39	1386.91	1385.78	1385.56
RC2	3.25	3.25	3.25	3.25	3.25	3.25	3.25	3.25	3.25
	1159.37	**1108.52**	1128.38	1124.46	1122.79	1124.77	1140.06	1123.49	1135.46
CNV	405	405	405	405	405	405	407.5	405	407.5
CTD	57952	**57192**	57710	57273	57444	57360	57641	57332	57550
CPU	P-400 Mhz	P-400 Mhz	P-200Mhz	SU 10	P3 1Ghz	P4 3Ghz	P4 3Ghz	P4 3Ghz	P4 3Ghz
T. (s)	1800	N/A	4950	7200	15000	86	86	146	146
Exp.	3	N/A	1	> 5	1	10	1	10	1

Table 1: Solomon instances with 100 customers. The table compares the ALNS heuristic to the heuristics by Berger et al. (BBB) [4], Homberger and Gehring (HG) [32], Bräysy (B) [7], Bent and Van Hentenryck (BH) [2] and Ibaraki et al. (IIKMUY) [33]. The data set is divided into six groups: R1, R2, C1, C2, RC1, RC2. For each group we report two numbers per heuristic. The top number is the number of vehicles used and the bottom number is the distance traveled. These numbers have been averaged over all the instance in the given group. The rows named *CNV* and *CTD* show the cumulative number of vehicles and distances respectively. The row *CPU* shows the computer used in the experiment and the row *T. (s)* shows the number of CPU seconds used for finding the solutions. The last row shows the number of experiments that were performed in order to obtain the results presented in the table (if multiple experiments were performed, the table shows the best results obtained). The two columns for the ALNS heuristic show the results obtained with the 25000 iteration configuration and the 50000 iteration configuration. For each configuration we show two columns. The first column shows the best result out of ten experiments, and the second column show the average solution quality (averaged over the ten experiments). Bold entries mark the best solution quality obtained among the heuristics in the comparison.

Customers	Instances	Solved to optimality	Optimums found	Avg. gap all (%)	Avg. gap opt.(%)	Avg. time (s)
25	56	56	56	0.02	0.02	5
50	56	53	48	0.19	0.13	15
100	56	37	27	0.36	0.26	47

Table 2: Comparison of ALNS to exact methods. The columns should be interpreted as follows: *Customers* — the number of customers in the test set, *Instances* — the number of instances in the test set, *Solved to optimality* — the number of instances that has been solved to optimality in the literature, *Optimums found* — the number of optimal solutions that were found by the heuristic, *Avg. gap all (%)* — the average gap over all instances, *Avg. gap opt. (%)* — the average gap over instances solved to optimality in the literature, *Avg. time (s)* — the average time in seconds spent on performing one experiment.

	GH99	GH01	BH	LL	LC	BHD	MB	ALNS 25K		ALNS 50K	
R1	18.2	18.2	18.2	18.3	18.2	18.2	**18.2**	18.2	18.20	18.2	18.20
	3705.00	3855.03	3677.96	3736.20	3676.95	3718.30	**3618.68**	3635.94	3664.648	3631.226	3652.747
R2	4.0	4.0	4.1	4.1	4.0	4.0	**4.0**	4.0	4.05	4.0	4.05
	3055.00	3032.49	3023.62	3023.00	2986.01	3014.28	**2942.92**	2950.30	2950.04	2949.368	2942.594
C1	18.9	18.9	18.9	19.1	18.9	18.9	**18.8**	18.9	18.90	18.9	18.90
	2782.00	2842.08	2726.63	2728.60	2743.66	2749.83	**2717.21**	2723.10	2732.458	2721.522	2728.382
C2	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.00	**6.0**	6.00
	1846.00	1856.99	1860.17	1854.90	1836.10	1842.65	1833.57	1833.33	1836.4	**1832.947**	1834.675
RC1	18.0	18.1	18.0	18.3	18.0	18.0	18.0	18.0	18.00	**18.0**	18.00
	3555.00	3674.91	3279.99	3385.80	3449.71	3329.62	3221.34	3233.76	3282.989	**3212.282**	3257.168
RC2	4.3	4.4	4.5	4.9	4.3	4.4	4.4	4.3	4.33	**4.3**	4.33
	2675.00	2671.34	2603.08	2518.70	2613.75	2585.89	2519.79	2560.59	2592.39	**2556.874**	2578.575
CNV	694	696	697	707	694	695	**694**	694	694.8	694	694.8
CTD	176180	179328	171715	172472	173061	172406	**168573**	169370	170589	169042	169941
CPU	P-200Mhz	P-400Mhz	SU 10	P-545Mhz	P-933Mhz	A-700Mhz	P4 2Ghz	P4-3Ghz	P4-3Ghz	P4-3Ghz	P4-3Ghz
T. (min)	4x10	4x2.1	n/a	182.1	5x10	2.4	8	4.3	4.3	7.7	7.7
Exp.	1	3	n/a	3	1	3	1	10	1	10	1

Table 3: Gehring/Homberger VRPTW instances with 200 customers. The table compares the ALNS heuristic to the heuristics by Gehring and Homberger (GH99) [27] and (GH01) [28], Bent and Van Hentenryck (BH) [2], Le Bouthillier and Cranic (LC) [5], Bräysy et al (BHD) [9] and Mester and Bräysy (MB) [42]. The table should be interpreted like Table 1. Notice that computing times are reported in minutes. Entries of the form $x \times y$ appearing in the *T. (min)* row indicate that the experiment was run for y minutes on a parallel computer with x processors.

	GH99	GH01	BH	LL	LC	BHD	MB	ALNS 25K		ALNS 50K	
R1	36.4	36.4	36.4	36.6	36.5	36.4	**36.3**	36.4	36.40	36.4	36.40
	8925.00	9478.22	8713.37	8912.40	8839.28	8692.17	**8530.03**	8609.38	8663.57	8540.04	8589.90
R2	8.0	8.0	8.0	8.0	8.0	8.0	**8.0**	8.0	8.00	8.0	8.00
	6502.00	6650.28	6959.75	6610.60	6437.68	6382.63	**6209.94**	6252.01	6309.84	6241.72	6277.07
C1	38.0	38.0	38.0	38.7	37.9	37.9	37.9	37.6	37.62	**37.6**	37.62
	7584.00	7855.82	7220.96	7181.40	7447.09	7230.48	7148.27	7369.88	7450.84	**7290.16**	7372.49
C2	12.0	12.0	12.0	12.1	12.0	12.0	**12.0**	12.0	12.00	12.0	12.00
	3935.00	3940.19	4154.40	4017.10	3940.87	3894.48	**3840.85**	3849.27	3884.44	3844.69	3875.95
RC1	36.1	36.1	36.1	36.5	36.0	36.0	**36.0**	36.0	36.00	36.0	36.00
	8763.00	9294.99	8330.98	8377.90	8652.01	8305.55	**8066.44**	8149.61	8240.28	8069.30	8148.81
RC2	8.6	8.8	8.9	9.5	8.6	8.9	8.8	8.5	8.64	**8.5**	8.64
	5518.00	5629.43	5631.70	5466.20	5511.22	5407.87	5243.06	5366.82	5388.76	**5335.09**	5351.56
CNV	1390	1392	1393	1414	1390	1391	1389	1385	1386.6	**1385**	1386.6
CTD	412270	428489	410112	405656	408281	399132	390386	395970	399377	**393210**	396158
CPU	P-200Mhz	P-400Mhz	SU 10	P-545Mhz	P-933Mhz	A-700Mhz	P4 2Ghz	P4-3Ghz	P4-3Ghz	P4-3Ghz	P4-3Ghz
T. (min)	4x20	4x7.1	n/a	359	5x20	7.9	17	9.7	9.7	15.8	15.8
Exp.	1	3	n/a	3	1	3	1	5	1	5	1

Table 4: Gehring/Homberger VRPTW instances with 400 customers

	GH99	GH01	BH	LL	LC	BHD	MB	ALNS 25K		ALNS 50K	
R1	54.5	54.5	55.0	55.2	54.8	54.5	**54.5**	54.5	54.50	54.5	54.50
	20854.00	21864.47	19308.62	19744.80	19869.82	19081.18	**18358.68**	19370.04	19562.34	18888.52	19048.49
R2	11.0	11.0	11.0	11.1	11.2	11.0	11.0	11.0	11.00	**11.0**	11.00
	13335.00	13656.15	14855.43	13592.40	13093.97	13054.83	12703.52	12729.51	12826.39	**12619.26**	12721.32
C1	57.9	57.7	57.8	58.2	57.9	57.8	57.8	57.5	57.56	**57.5**	57.56
	14792.00	14817.25	14357.11	14267.30	14205.58	14165.90	14003.09	14125.94	14212.71	**14065.89**	14098.04
C2	17.9	17.8	17.8	18.2	17.9	18.0	17.8	17.5	17.80	**17.5**	17.80
	7787.00	7889.96	8259.04	8202.60	7743.92	7528.73	7455.83	7891.70	7834.72	**7801.296**	7682.61
RC1	55.1	55.0	55.1	55.5	55.2	55.0	**55.0**	55.0	55.00	55.0	55.00
	18411.00	19114.02	17035.91	17320.00	17678.13	16994.22	**16418.63**	16846.71	17006.94	16594.94	16722.51
RC2	11.8	11.9	12.4	13.0	11.8	12.1	12.1	11.6	11.78	**11.6**	11.78
	11522.00	11670.29	11987.89	11204.90	11034.71	11212.36	10677.46	10922.44	10938.30	**10777.12**	10828.45
CNV	2082	2079	2091	2112	2088	2084	2082	2071	2076.4	**2071**	2076.4
CTD	867010	890121	858040	843320	836261	820372	796172	818863	823814	**807470**	811014
CPU	P-200Mhz	P-400Mhz	SU 10	P-545Mhz	P-933Mhz	A-700Mhz	P4 2Ghz	P4-3Ghz	P4-3Ghz	P4-3Ghz	P4-3Ghz
T. (min)	4x30	4x12.9	n/a	399.8	5x30	16.2	40	10.5	10.5	18.3	18.3
Exp.	1	3	n/a	3	1	3	1	5	1	5	1

Table 5: Gehring/Homberger VRPTW instances with 600 customers.

	GH99	GH01	BH	LL	LC	BHD	MB	ALNS 25K		ALNS 50K	
R1	72.8	72.8	**72.7**	73.0	73.1	72.8	72.8	72.8	72.80	72.8	72.80
	34586.00	34653.88	**33337.91**	33806.34	33552.40	32748.06	31918.47	32697.85	32905.52	32316.79	32528.76
R2	15.0	15.0	15.0	15.1	15.0	15.0	**15.0**	15.0	15.00	15.0	15.00
	21697.00	21672.85	24554.63	21709.39	21157.56	21170.15	**20295.28**	20477.77	20627.40	20353.51	20499.72
C1	76.7	76.1	76.1	77.4	76.3	76.3	76.2	75.6	75.66	**75.6**	75.66
	26528.00	26936.68	25391.67	25337.02	25668.82	25170.88	25132.27	25365.59	25547.82	**25193.13**	25269.64
C2	24.0	23.7	24.4	24.4	24.1	24.2	**23.7**	23.7	23.98	23.7	23.94
	12451.00	11847.92	14253.83	11956.60	11985.11	11648.92	**11352.29**	11985.80	11999.28	11725.46	11741.73
RC1	72.4	72.3	73.0	73.2	**72.3**	73.0	73.0	73.0	73.00	73.0	73.00
	38509.00	40532.35	30500.15	31282.54	**37722.62**	30005.95	30731.07	29864.06	30016.05	29478.3	29625.04
RC2	16.1	16.1	16.6	17.1	15.8	16.3	15.8	15.7	15.82	**15.7**	15.82
	17741.00	17941.23	18940.84	17561.22	17441.60	17686.65	16729.18	16870.87	17022.33	**16761.95**	16852.95
CNV	2770	2760	2778	2802	2766	2776	2765	2758	2762.6	**2758**	2762.2
CTD	1515120	1535849	1469790	1416531	1475281	1384306	1361586	1372619	1381184	**1358291**	1365178
CPU	P-200Mhz	P-400Mhz	SU 10	P-545Mhz	P-933Mhz	A-700Mhz	P4-2Ghz	P4-3Ghz	P4-3Ghz	P4-3Ghz	P4-3Ghz
T. (min)	4x40	4x23.2	n/a	512.9	5x40	26.2	145	13.5	13.5	22.7	22.7
Exp.	1	3	n/a	3	1	3	1	5	1	5	1

Table 6: Gehring/Homberger VRPTW instances with 800 customers

	GH99	GH01	BH	LL	LC	BHD	MB	ALNS 25K		ALNS 50K	
R1	**91.9**	91.9	92.8	92.7	92.2	92.1	92.1	92.2	92.30	92.2	92.30
	57186.00	58069.61	51193.47	50990.80	55176.95	50025.64	49281.48	52131.96	51900.53	50751.25	50584.55
R2	19.0	19.0	19.0	19.0	19.2	19.0	19.0	19.0	19.00	**19.0**	19.00
	31930.00	31873.62	36736.97	31990.90	30919.77	31458.23	29860.32	30108.84	30327.34	**29780.82**	30016.08
C1	96.0	95.4	95.1	96.3	95.3	95.8	95.1	94.6	94.72	**94.6**	94.72
	43273.00	43392.59	42505.35	42428.50	43283.92	42086.77	41569.67	42123.87	42266.42	**41877.00**	42034.65
C2	30.2	29.7	30.3	30.8	29.9	30.6	**29.7**	29.7	29.90	29.7	29.86
	17570.00	17574.72	18546.13	17294.90	17443.50	17035.88	**16639.54**	17307.16	17589.70	16840.37	17052.62
RC1	90.0	90.1	90.2	90.4	90.0	90.0	**90.0**	90.0	90.00	90.0	90.00
	50668.00	50950.14	48634.15	48892.40	49711.36	46736.92	**45396.41**	47735.43	48168.74	46752.15	47081.64
RC2	19.0	18.5	19.4	19.8	18.5	19.0	18.7	18.3	18.46	**18.3**	18.46
	27012.00	27175.98	29079.78	26042.30	26001.11	25994.12	25063.51	25267.93	25466.13	**25090.88**	25185.45
CNV	3461	3446	3468	3490	3451	3465	3446	3438	3443.8	**3438**	3443.4
CTD	2276390	2290367	2266959	2176398	2225366	2133376	2078110	2146752	2157189	**2110925**	2119550
CPU	P-200Mhz	P-400Mhz	SU 10	P-545Mhz	P-933Mhz	A-700Mhz	P4 2Ghz	P4-3Ghz	P4-3Ghz	P4-3Ghz	P4-3Ghz
T. (min)	4x50	4x30.1	n/a	606.3	5x50	39.6	600	16	16	26.6	26.6
Exp.	1	3	n/a	3	1	3	1	5	1	5	1

Table 7: Gehring/Homberger VRPTW instances with 1000 customers.

	R1		R2		C1		C2		RC1		RC2	
#	Veh.	Dist.	Veh.	Dist.	Veh.	Dist.	Veh.	Dist.	Veh.	Dist.	Veh.	Dist.
200 customers												
1	20	4785.96	4	4563.55	**20**	**2704.57**	**6**	**1931.44**	**18**	**3647.56**	6	3126.03
2	18	4059.57	4	3650.54	18	2943.83	**6**	**1863.16**	**18**	**3269.91**	5	2828.39
3	18	3387.64	**4**	**2892.07**	18	2710.21	6	1776.96	18	3034.45	**4**	**2613.12**
4	18	3086.11	4	1981.30	18	2644.92	**6**	**1713.46**	**18**	**2869.74**	**4**	**2052.74**
5	18	4125.19	4	3377.18	**20**	**2702.05**	**6**	**1878.85**	18	3430.03	**4**	**2912.13**
6	**18**	**3586.80**	4	2929.72	**20**	**2701.04**	**6**	**1857.35**	**18**	**3357.90**	**4**	**2975.13**
7	18	3160.44	4	2456.71	**20**	**2701.04**	**6**	**1849.46**	**18**	**3233.29**	**4**	**2539.85**
8	18	2971.66	**4**	**1849.87**	19	2775.48	**6**	**1820.53**	**18**	**3110.46**	**4**	**2314.61**
9	18	3802.55	4	3113.74	18	2687.83	**6**	**1830.05**	18	3114.02	4	2175.98
10	18	3312.44	4	2666.10	**18**	**2644.25**	6	1808.21	**18**	**3020.24**	4	2015.61
400 customers												
1	40	10432.30	8	9338.49	40	7152.06	12	4116.33	**36**	**8813.43**	**11**	**6834.02**
2	**36**	**9115.68**	**8**	**7649.87**	**36**	**7733.55**	**12**	**3930.05**	**36**	**8118.43**	**9**	**6355.59**
3	36	7988.22	8	5998.04	**36**	**7082.13**	12	3775.32	**36**	**7663.73**	**8**	**5055.02**
4	36	7415.81	**8**	**4326.48**	**36**	**6816.17**	12	3543.60	**36**	**7368.47**	**8**	**3647.39**
5	**36**	**9479.10**	8	7252.64	40	7152.06	12	3946.14	**36**	**8426.57**	9	6119.44
6	36	8556.38	8	6212.37	40	7153.45	**12**	**3875.94**	36	8390.24	**8**	**5997.24**
7	36	7725.97	8	5136.74	**39**	**7546.78**	12	3894.98	36	8223.65	**8**	**5476.57**
8	**36**	**7390.76**	**8**	**4055.22**	**37**	**7546.32**	12	3796.00	36	7922.67	8	4877.39
9	36	8970.98	8	6507.40	36	7573.18	12	3881.21	36	7953.20	**8**	**4601.30**
10	36	8325.16	**8**	**5894.40**	36	7145.92	12	3687.13	36	7774.83	8	4355.52
600 customers												
1	59	21677.41	11	18837.28	**60**	**14095.64**	18	7780.84	55	17751.33	**15**	**13163.03**
2	54	20045.49	**11**	**15069.24**	**56**	**14174.12**	**17**	**8799.38**	55	16548.43	**12**	**11853.72**
3	54	17733.91	**11**	**11291.52**	**56**	**13803.50**	**17**	**7604.00**	**55**	**15499.02**	**11**	**9863.35**
4	54	16374.29	**11**	**8163.24**	**56**	**13578.66**	**17**	**6993.77**	**55**	**15072.90**	**11**	**7231.64**
5	54	21243.24	**11**	**15418.00**	60	14085.72	18	7578.12	**55**	**17401.34**	12	12560.43
6	54	18948.53	**11**	**12936.28**	**60**	**14089.66**	18	7554.61	55	17355.10	**11**	**12282.52**
7	54	17438.28	**11**	**10269.96**	**58**	**15017.03**	**18**	**7520.34**	55	17058.40	**11**	**11052.49**
8	54	16146.17	**11**	**7752.78**	**57**	**14343.05**	**17**	**8696.15**	55	16510.65	11	10488.75
9	54	20375.70	11	13885.52	56	13767.45	18	7356.19	55	16435.71	**11**	**9882.71**
10	54	18902.19	**11**	**12568.79**	**56**	**13688.57**	17	7938.94	55	16316.51	11	9340.06
800 customers												
1	80	37492.04	15	28822.48	80	25184.38	24	11664.00	**73**	**31275.38**	**19**	**20954.95**
2	72	33816.69	**15**	**23274.22**	**74**	**25536.76**	24	11428.07	73	29172.08	**17**	**18032.89**
3	72	30317.49	15	18078.82	**72**	**24629.86**	24	11184.67	73	28164.66	**15**	**14800.78**
4	72	28568.78	**15**	**13413.79**	**72**	**23938.33**	23	10999.42	73	27201.39	**15**	**11368.19**
5	72	35503.63	15	25077.09	**80**	**25166.28**	24	11451.57	73	30548.23	16	19180.13
6	72	32360.07	15	20969.81	**80**	**25160.85**	24	11403.57	73	30511.07	15	19075.89
7	72	29979.63	**15**	**16977.49**	79	25425.92	24	11412.08	73	30007.82	**15**	**17329.32**
8	**72**	**28341.21**	**15**	**12945.52**	**75**	**25450.99**	**23**	**13878.40**	73	29547.96	**15**	**16226.78**
9	**72**	**34218.41**	15	22877.21	**72**	**25737.46**	24	11650.10	73	29360.93	**15**	**15687.20**
10	72	32569.97	**15**	**21092.27**	**72**	**25697.68**	23	12103.56	73	28993.52	15	14944.14
1000 customers												
1	100	54720.19	19	43264.68	**100**	**42478.95**	**30**	**16879.24**	90	48933.68	**21**	**30396.13**
2	**91**	**55428.79**	**19**	**34417.47**	**91**	**42249.60**	29	17563.06	90	46165.33	**18**	**27552.05**
3	91	49634.84	**19**	**25400.16**	**90**	**40376.43**	30	16109.71	90	44014.81	18	20811.18
4	91	45303.47	**19**	**18332.77**	**90**	**39980.07**	**29**	**16011.30**	90	42607.34	**18**	**16007.59**
5	92	53089.15	19	37746.01	**100**	**42469.18**	30	16596.69	90	48934.53	**18**	**28368.48**
6	91	54555.32	19	30778.85	**100**	**42471.29**	**30**	**16369.10**	90	48766.98	18	28746.61
7	91	48141.47	**19**	**23991.71**	**99**	**42673.51**	31	16590.48	90	48005.94	18	26765.43
8	91	44853.70	**19**	**17844.36**	**95**	**42359.27**	29	18407.27	90	47122.61	**18**	**24961.29**
9	92	52015.72	19	34349.70	**91**	**41482.00**	**30**	**16294.72**	90	46889.79	18	24113.72
10	92	49769.85	19	31682.52	90	42214.60	29	17582.15	90	46080.51	18	23056.75

Table 8: The table shows the best solutions to large VRPTW instances identified by the ALNS heuristic. The first column shows the problem number. The columns *veh.* and *dist.* show the number of vehicles and total distance traveled in the best solution found. The table is grouped by instance type and instance size. Bold entries indicate a best solution (either a tie with one of the heuristics from the literature or a new best solution).

are compared to the best results obtained by heuristics proposed by Chao et al. [11], Renaud et al. [48] and Cordeau et al. [17]. The heuristic that previously has achieved the best solution quality is the one proposed by Cordeau et al. The cost of a solution is defined as the total distance traveled by the vehicles. The table shows that the ALNS heuristic has been able to improve upon the best solution for a considerable number of instances. Each configuration has found 14 new best solutions, but as most of these overlap, the total number of new best solutions is 15. The individual improvements are typically rather small though. The table also shows that the ALNS heuristic is quite stable as the average gap from the best known solution never surpasses 2% and 1% in the ALNS-25K and ALNS-50K configurations, respectively. It should be mentioned that the ALNS heuristic is slower than the previously proposed heuristics. The ALNS-25K and ALNS-50K configurations use on average two and four minutes respectively to perform one experiment on a 3GHz Pentium 4. The heuristic by Cordeau et al. on average used 11.7 minutes to perform one experiment on a Sun SPARCstation 10 which is considerably slower than our computer.

6.3.3 Site dependent vehicle routing problem (SDVRP)

The heuristic has been applied to the same test instances as used by [18]. The results obtained on the SDVRP instances are summarized in Table 10. The results are promising as the average solution quality of ALNS-25K overall is better than results previously published. Also the sum of the costs of the best known solutions found by the ALNS-50K configuration is more than 2% better than the previous best known solution and the best known solution was improved for 30 out of the 35 instances. The computational time needed for performing one experiment with the ALNS-25K configuration seems to be roughly comparable with the time needed for performing one experiment with the heuristic proposed by Cordeau and Laporte [18]. The ALNS-25K configuration spends on average 1.4 minutes to perform one experiment while the heuristic by Cordeau and Laporte spent around 12 minutes to perform the same task on a Sun Ultra 2, 300 MHz. It should be mentioned that the problem PR02 caused the ALNS heuristic some difficulties, as it was only able to find a feasible solution in one out of ten experiments for the ALNS-25K configuration and three out of ten experiments for the ALNS-50K configuration.

6.3.4 Capacitated vehicle routing problem (CVRP)

For the CVRP we have chosen to test the ALNS heuristic on three datasets. The first dataset was proposed by Christofides et al. [13] and contains instances with between 50 and 200 customers. The second dataset was proposed by Golden et al. [30] and contains instances with up to 483 customers. The last dataset was proposed by Li et al. [40] and contains instances with up to 1200 customers. These are the so-far largest instances that the ALNS heuristic has been applied to. Table 11 summarizes these experiments. Notice that we only compare the ALNS heuristic to a subset of all the CVRP heuristics that have been proposed in the literature. The heuristics used for benchmarking are the most recent heuristics that were surveyed by Cordeau et al. [16].

The table shows that the ALNS heuristic cannot compete with the well-performing heuristic by Mester and Bräysy [42], but its performance is comparable to the rest of the heuristics. For the last dataset, the heuristic proposed by Li et al. must be considered to be the best as it is very fast compared to the ALNS heuristic although the ALNS heuristic overall is able to reach better solutions. We discovered one new best solution for the Golden et al. dataset and three new best solutions for the Li et al. dataset.

6.3.5 Open vehicle routing problem (OVRP)

The results on the OVRP are summarized in Table 12. The heuristic was tested on the same 16 instances that were used by Brandão [6] and Fu et al. [25]. The primary objective considered was to minimize the number of vehicles used, while the secondary objective was to minimize the traveled distance. The solutions obtained by the ALNS heuristic are promising as the best known solution to 11 out of the 16 instances has been improved. The running time of the ALNS heuristic is comparable to the two other heuristics: The configuration of Brandão's heuristic that obtains the best results spends on average 9.6 minutes to solve an instance on a 500MHz Pentium III. In the paper by Fu et al. two configurations of their heuristic are tested. These configurations spend on average 6.6 and 13.9 minutes respectively to solve an instance on a 600 MHz Pentium II. The ALNS-25K and ALNS-50K configurations use 1.4 and 2.3 minutes respectively to solve an instance on a 3GHz Pentium IV.

6.3.6 Computational results conclusion

The computational results presented in this section are very encouraging. The results show that the general ALNS heuristic is on par with the best specialized heuristics for the VRPTW and that the heuristic currently is the best when it comes to minimizing the number of vehicles in large VRPTW instances. One should keep in mind that numerous specialized heuristics have been proposed for the VRPTW making it difficult for a general heuristic to compete on these instances.

For the MDVRP, SDVRP and OVRP the ALNS heuristic has been able to find many new best solutions and the results on the SDVRP are especially promising. For the CVRP the proposed heuristic is able to compete with many of the most recent heuristics, but it is outperformed by a more specialized heuristic for this problem. Nevertheless, a couple of new best solutions were found for this problem type also. One should also keep in mind that the heuristic was not tuned for each problem type, but a general parameter setting was used for all experiments.

The comparison between the fast and the slow version of the ALNS heuristic showed that it did not pay off to use the ALNS-50K variant for the smaller instances, while for instances with around 400 to 600 or more customers it seemed worthwhile to use the ALNS-50K configuration. Consequently, it might be useful to use a variable number of iterations I which depends on the number n of requests. E.g. $I := 20000 + 50n$.

7 Conclusion

A new general heuristic framework, denoted Adaptive Large Neighborhood Search has been presented. The framework has been used to solve several variants of vehicle routing problems in the present paper as well as in [49, 50]. This includes the vehicle routing problem with time windows (VRPTW), the capacitated vehicle routing problem (CVRP), the multi-depot vehicle routing problem (MDVRP), the site dependent vehicle routing problem (SDVRP), the open vehicle routing problem (OVRP), the pickup and delivery problem with time windows (PDPTW), the vehicle routing problem with backhauls (VRPB), the mixed vehicle routing problem with backhauls (MVRPB), the multi-depot mixed vehicle routing problem with backhauls (MDMVRPB), the vehicle routing problem with backhauls and time windows (VRPBTW), the mixed vehicle routing problem with backhauls and time windows (MVRPBTW) and the vehicle routing problem with simultaneous deliveries and pickups (VRPSDP).

Due to the generality of the ALNS framework and the encouraging results demonstrated for a wide spectrum of VRP problems, we believe that ALNS should be considered as one of the standard frameworks for solving large-sized optimization problems.

Supply chain management is a research area getting increasing attention [37]. By co-ordinating activities in the supply chain, companies can rationalize the process resulting in mutual gains. If the involved companies co-ordinate their transportation activities we will see a need for solving mixed transportation problems, where the instances for example consist of a mixture of PDPTW, MDVRP and SDVRP problems. In order to handle future changes in the distribution structure, these algorithms need to be stable for various input types, and should not need to be tuned for particular problem characteristics. It should be clear that the ALNS framework is very promising for these such types.

In conclusion we may add an interesting observation: We have seen that a mixture of good and less good heuristics lead to better solutions than using good heuristics solely. It is however necessary to hierarchically control the search, such that well-performing heuristics are given most influence, but such that all heuristics participate in the solution process. Using this principle one gets a robust and well-performing solution approach.

8 Acknowledgments

The authors wish to thank the anonymous referee for valuable comments and corrections.

9 Appendix

New best solution to the Solomon R207 instance

Route	Length	Visit sequence
1	437.339	42 92 45 46 36 64 11 62 88 30 20 65 71 9 81 34 78 79 3 76 28 53 40 2 87 57 41 22 73 21 72 74 75 56 4 25 55 54 80 68 77 12 26 58 13 97 37 100 98 93 59 95 94
2	453.269	27 1 69 50 33 29 24 39 67 23 15 43 14 44 38 86 16 61 91 85 99 96 6 84 8 82 7 48 47 49 19 10 63 90 32 66 35 51 70 31 52 18 83 17 5 60 89

Total length: 890.61

Full VRPTW tables

The full tables documenting the VRPTW experiments described in section 6.3.1 can be found in Tables 13 – 18.

Tables 19 – 21 contain detailed result from the experiment comparing the ALNS heuristic to exact methods.

Full CVRP tables

The detailed results for the CVRP experiments described in section 6.3.4 can be found in Tables 22 – 24.

References

[1] R. K. Ahuja, Ö. Ergun, J. b. Orlin, and A. P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123:72–102, 2002.

[2] R. Bent and P. V. Hentenryck. A two-stage hybrid local search for the vehicle routing problem with time windows, 2004. To appear in Transportation Science.

[3] J. Berger and M. Barkaoui. A new hybrid genetic algorithm for the capacitated vehicle routing problem. *Journal of the Operational Research Society*, 54:1254–1262, 2003.

[4] J. Berger, M. Barkaoui, and O. Bräysy. A route hybrid genetic approach for the vehicle routing problem with time windows. *INFOR*, 41(2), 2003.

[5] A. L. Bouthillier and T. G. Crainic. A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. *Computers and Operations Research*, 2004. To appear.

[6] J. Brandão. A tabu search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, 157(3):552–564, 2004.

[7] O. Bräysy. A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing*, 15(4):347–368, 2003.

[8] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*. To appear.

[9] O. Bräysy, G. Hasle, and W. Dullaert. A multi-start local search algorithm for the vehicle routing problem with time windows. *European Journal of Operational Research*, 2003. To appear.

[10] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation, 2003. Working Paper, ILOG, Madrid, 2003.

[11] I. Chao, B. L. Golden, and E. Wasil. A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. *Am. J. Math. Mgmt. Sci.*, 13:371–406, 1993.

[12] I.-M. Chao, B. Golden, and E. Wasil. A computational study of a new heuristic for the site-dependent vehicle routing problem. *INFOR*, 37(3):319–336, 1999.

[13] N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, chapter 11, pages 315 – 338. John Wiley & Sons, 1979.

[14] W. Cook and J. L. Rich. A parallel cutting-plane algorithm for the vehicle routing problem with time windows. Technical Report TR99-04, Departement of Computational and Applied Mathematics, Rice University, 1999.

[15] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis. Vrp with time windows. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 7, pages 157–193. SIAM, Philidelphia, 2002.

[16] J.-F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, and J.-S. Sormany. New heuristics for the vehicle routing problem. Technical Report G-2004-33, GERAD, Montreal, Canada, 2004.

[17] J.-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30:105–119, 1997.

[18] J.-F. Cordeau and G. Laporte. A tabu search algorithm for the site dependent vehicle routing problem with time windows. *INFOR*, 39:292–298, 2001.

[19] J.-F. Cordeau, G. Laporte, and A. Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52:928–936, 2000.

[20] E. Danna and C. L. Pape. Accelerating branch-and-price with local search: A case study on the vehicle routing problem with time windows. Technical Report 03-006, ILOG, ILOG S.A, 9, rue de Verdun, F-94253 Gentilly Cédex, Septemper 2003.

[21] W. A. Dees, Jr. and P. G. Karger. Automated rip-up and reroute techniques. In *Proceedings of the 19th conference on Design automation*, pages 432–439. IEEE Press, 1982.

[22] G. Desaulniers, J. Desrosiers, A. Erdmann, M. M. Solomon, and F. Soumis. Vrp with pickup and delivery. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 9, pages 225–242. SIAM, Philidelphia, 2002.

[23] O. Ergun, J. B. Orlin, and A. Steele-Feldman. Creating very large scale neighborhoods out of smaller ones by compounding moves: a study on the vehicle routing problem, 2003. Working Paper, Massachusetts Institute of Technology.

[24] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

[25] Z. Fu, R. W. Eglese, and L. Li. A tabu search heuristic for the open vehicle routing problem. Technical Report 2003/042, Lancaster University Management School, Lancaster, United Kingdom, 2003.

[26] R. Fukasawa, J. Lysgaard, M. P. de Aragão, M. Reis, E. Uchoa, and R. F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. In *Proceedings of IPCO X*. Columbia University, 2004.

[27] H. Gehring and J. Homberger. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In K. Miettinen, M. Mäkelä, and J. Toivanen, editors, *Proceedings of EUROGEN99 – Short Course on Evolutionary Algorithms in Engineering and Computer Science*, pages 57–64. University of Jyväskylä, 1999.

[28] H. Gehring and J. Homberger. A parallel two-phase metaheuristic for routing problems with time windows. *Asia-Pacific Journal of Operational Research*, 18:35–47, 2001.

[29] M. Gendreau, G. Laporte, and J.-Y. Potvin. Metaheuristics for the capacitated vrp. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 6, pages 129–154. SIAM, Philidelphia, 2002.

[30] B. L. Golden, E. A. Wasil, J. P. Kelly, and I.-M. Chao. Metaheuristics in vehicle routing. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 33–56. Kluwer, Boston (MA), 1998.

[31] P. Hansen and N. Mladenovic. An introduction to variable neighborhood search. In S. V. et al., editor, *Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization*, pages 433–458. Kluwer Academic Publishers, Dordrecht, 1999.

[32] J. Homberger and H. Gehring. A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 2004. To appear.

[33] T. Ibaraki, S. Imahori, T. Masuda, T. Uno, and M. Yagiura. A route hybrid genetic approach for the vehicle routing problem with time windows. *Transportation Science*. To appear.

[34] S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. Technical Report G-2003-55, GERAD, Montreal, Canada, September 2003.

[35] B. Kallehauge, J. Larsen, and O. B. G. Madsen. Lagrangean duality applied on vehicle routing with time windows - experimental results. Technical Report IMM-REP-2000-8, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2001.

[36] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101 – 116, 1999.

[37] D. Lambert and M. Cooper. Issues in supply chain management. *Marketing Management*, 29:65–83, 2000.

[38] G. Laporte and F. Semet. Classical heuristics for the capacitated vrp. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 5, pages 109–128. SIAM, Philidelphia, 2002.

[39] J. Larsen. *Parallellization of the Vehicle Routing Problem with Time Windows*. Ph.D. thesis, Department of Mathematical Modelling, Technical University of Denmark, 1999. IMM-PHD-1999-62.

[40] F. Li, B. Golden, and E. Wasil. Very large-scale vehicle routing: new test problems, algorithms, and results. *Computers and Operations Research*, 2004. To appear.

[41] J. Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming, Ser. A*, 100:423–445, 2004.

[42] D. Mester and O. Bräysy. Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers and Operations Research*, 2004. To appear.

[43] B. Nag, B. L. Golden, and A. Assad. Vehicle routing with site dependencies. In B. Golden and A. Assad, editors, *Vehicle Routing: Methods and Studies*, pages 149–159. Elsevier Science Publishers B.V, 1988.

[44] D. Pisinger. Upper bounds and exact algorithms for p-dispersion problems. *Computers and Operations Research*, 2004. To appear.

[45] J.-Y. Potvin and J.-M. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66:331–340, 1993.

[46] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers and Operations Research*, 31:1985–2002, 2004.

[47] M. Reimann, K. Doerner, and R. Hartl. D-ants: Savings based ants divide and conquer the vrp. *Computers and Operations Research*, 31:563–591, 2004.

[48] J. Renaud, G. Laporte, and F. F. Boctor. A tabu search heuristic for the multi-depot vehicle routing problem. *Computers and Operations Research*, 23(3):229–235, 1996.

[49] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows, 2004. Submitted to Transportation Science.

[50] S. Ropke and D. Pisinger. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 2004. To appear.

[51] D. Sariklis and S. Powell. A heuristic method for the open vehicle routing problem. *Journal of the Operational Research Society*, 51:564–573, 2000.

[52] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171, 2000.

[53] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming)*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. 1998.

[54] M. Sigurd, D. Pisinger, and M. Sig. The pickup and delivery problem with time windows and precedences. *Transportation Science*, 38:197–209, 2004.

[55] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

[56] E. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23:661–673, 1993.

[57] C. Tarantilis and C. Kiranoudis. Boneroute: an adaptive memory-based method for effective fleet management. *Annals of Operations Research*, 115:227–241, 2002.

[58] P. Toth and D. Vigo. The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing*, 15(4):333–346, 2003.

[59] M. A. Trick. A linear relaxation heuristic for the generalized assigment problem. *Naval Research Logistics*, 39:137–152, 1992.

	Best known					ALNS 25K				ALNS 50K			
	n	t	type	cost	ref	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
P01	50	4	C	**576.87**	CGW	576.87	**576.87**	0.00	14	576.87	**576.87**	0.00	29
P02	50	4	C	**473.53**	RLB	473.53	**473.53**	0.00	14	473.53	**473.53**	0.00	28
P03	75	2	C	**641.19**	CGW	641.19	**641.19**	0.00	32	641.19	**641.19**	0.00	64
P04	100	2	C	1001.59	CGL	1008.49	1001.59	0.74	42	1006.09	**1001.04**	0.50	88
P05	100	2	C	**750.03**	CGL	753.04	751.86	0.40	58	752.34	751.26	0.31	120
P06	100	3	C	**876.5**	RLB	884.36	880.42	0.90	47	883.01	876.70	0.74	93
P07	100	4	C	885.8	CGL	889.14	**881.97**	0.81	43	889.36	**881.97**	0.84	88
P08	249	2	CD	4437.68	CGL	4426.86	**4387.38**	0.90	166	4421.03	4390.80	0.77	333
P09	249	3	CD	3900.22	CGL	3902.18	3874.75	0.74	182	3892.50	**3873.64**	0.49	361
P10	249	4	CD	3663.02	CGL	3676.93	3655.18	0.74	180	3666.85	**3650.04**	0.46	363
P11	249	5	CD	3554.18	CGL	3592.82	3552.27	1.32	174	3573.23	**3546.06**	0.77	357
P12	80	2	C	**1318.95**	RLB	1319.70	**1318.95**	0.06	38	1319.13	**1318.95**	0.01	75
P13	80	2	CD	**1318.95**	RLB	1321.10	**1318.95**	0.16	30	1318.95	**1318.95**	0.00	60
P14	80	2	CD	**1360.12**	CGL	1360.12	**1360.12**	0.00	29	1360.12	**1360.12**	0.00	58
P15	160	4	C	**2505.42**	CGL	2517.96	**2505.42**	0.50	125	2519.64	**2505.42**	0.57	253
P16	160	4	CD	**2572.23**	RLB	2577.28	**2572.23**	0.20	92	2573.95	**2572.23**	0.07	188
P17	160	4	CD	**2709.09**	CGL	2709.65	**2709.09**	0.02	90	2709.09	**2709.09**	0.00	179
P18	240	6	C	**3702.85**	CGL	3751.85	3727.58	1.32	209	3736.53	**3702.85**	0.91	419
P19	240	6	CD	**3827.06**	RLB	3846.35	3839.36	0.50	158	3838.76	**3827.06**	0.31	315
P20	240	6	CD	**4058.07**	CGL	4065.32	**4058.07**	0.18	151	4064.76	**4058.07**	0.16	300
P21	360	9	C	**5474.84**	CGL	5576.82	5519.47	1.86	293	5501.58	**5474.84**	0.49	582
P22	360	9	CD	**5702.16**	CGL	5731.10	5714.46	0.51	228	5722.19	**5702.16**	0.35	462
P23	360	9	CD	6095.46	CGL	6107.84	**6078.75**	0.48	223	6092.66	**6078.75**	0.23	443
PR01	48	4	CD	**861.32**	CGL	861.32	**861.32**	0.00	16	861.32	**861.32**	0.00	30
PR02	96	4	CD	1307.61	CGL	1311.54	**1307.34**	0.32	52	1308.17	**1307.34**	0.06	103
PR03	144	4	CD	1806.6	CGL	1810.90	**1806.53**	0.24	106	1810.66	1806.60	0.23	214
PR04	192	4	CD	2072.52	CGL	2080.55	2066.64	0.95	146	2073.16	**2060.93**	0.59	296
PR05	240	4	CD	2385.77	CGL	2352.59	2341.65	0.63	188	2350.31	**2337.84**	0.53	372
PR06	288	4	CD	2723.27	CGL	2695.15	**2685.35**	0.36	232	2695.74	2687.60	0.39	465
PR07	72	6	CD	**1089.56**	CGL	1089.56	**1089.56**	0.00	29	1089.56	**1089.56**	0.00	58
PR08	144	6	CD	1666.6	CGL	1677.31	1665.80	0.75	105	1675.74	**1664.85**	0.65	207
PR09	216	6	CD	2153.1	CGL	2148.85	**2136.42**	0.58	173	2144.84	**2136.42**	0.39	350
PR10	288	6	CD	2921.85	CGL	2913.34	**2889.49**	0.83	228	2905.43	2889.82	0.55	455
Tot.				80394		80651.59	80249.57		3894	80448.26	80133.89		7809
Avg.								0.52	118			0.34	237
< PB							14				14		
#B				18			20				27		

Table 9: Multi depot vehicle routing problems. The leftmost column shows the problem name, while the rest of the table is divided into three major columns that display the previously best known results and the results obtained by the ALNS-25K and ALNS-50K configurations. The sub columns should be interpreted like this: *n* — number of customers, *t* — number of depots, *type* — the type of the instance (*C* indicates that the instance is capacity constrained, while *D* indicates that route duration constraints are present), *cost* — the cost of the previously best known solution (the cost is calculated as the total distance traveled), *ref* — where the solution was first reported. The following abbreviations are used: *CGW* — Chao et al. [11], *RLB* — Renaud et al. [48], *CGL* — Cordeau et al. [17]. The last 10 instances were introduced by Cordeau et al. [17] and the two other heuristics have not been applied to these instances. The columns *best sol.* and *avg. sol.* show the cost of the best solution and the average cost of the solutions obtained during 10 experiments. *avg. gap* shows how far the average solution cost is from the best known solution. *avg. time* shows how much time the heuristic spends in one experiment. The rows *Tot.* and *Avg.* sums and averages key columns. *"<PB"* shows how many times the best solution found by the ALNS configuration was better than the previous best known solution and *#B* shows the number of best known solutions obtained. Entries written in bold indicate best known solutions.

	Best known				ALNS 25K				ALNS 50K			
	n	t	cost	ref	avg. sol.	best sol.	avg. gap (%)	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
P01	50	3	642.66	CL	645.04	**640.32**	0.74	10	642.93	**640.32**	0.41	20
P02	50	2	**598.1**	CL	599.40	**598.10**	0.22	10	598.82	**598.10**	0.12	19
P03	75	3	959.36	CL	962.36	958.14	0.56	20	963.14	**957.04**	0.64	40
P04	75	2	**854.43**	CL	858.05	**854.43**	0.42	18	856.22	**854.43**	0.21	36
P05	100	3	1020.22	CL	1012.46	1007.51	0.89	34	1009.08	**1003.57**	0.55	68
P06	100	2	1036.02	CL	1034.09	1028.70	0.54	35	1032.67	**1028.52**	0.40	69
P07	27	3	**391.3**	CGW	391.30	**391.30**	0.00	4	391.30	**391.30**	0.00	8
P08	54	3	**664.46**	CGW	664.46	**664.46**	0.00	12	664.46	**664.46**	0.00	24
P09	81	3	**948.23**	CGW	958.69	**948.23**	1.10	24	961.36	**948.23**	1.38	47
P10	108	3	1223.88	CL	1229.42	**1218.75**	0.88	38	1225.28	**1218.75**	0.54	76
P11	135	3	1464.98	CL	1488.28	1468.38	1.70	58	1475.85	**1463.33**	0.86	116
P12	162	3	1695.67	CL	1697.98	1690.56	1.17	78	1689.62	**1678.40**	0.67	157
P13	54	3	1196.73	CL	1194.40	**1194.18**	0.02	12	1194.91	**1194.18**	0.06	24
P14	108	3	1962.66	CL	1961.11	**1960.62**	0.02	36	1960.83	**1960.62**	0.01	72
P15	162	3	2751.45	CL	2712.10	2695.22	1.01	77	2701.61	**2685.09**	0.61	152
P16	216	3	3491.18	CL	3421.74	3402.94	0.75	109	3411.50	**3396.36**	0.45	213
P17	270	3	4230.96	CL	4109.62	**4084.92**	0.60	146	4114.26	4085.61	0.72	291
P18	324	3	4929.71	CL	4821.55	4775.35	1.39	177	4795.31	**4755.50**	0.84	346
P19	100	3	850.39	CL	852.09	846.35	0.71	43	848.54	**846.07**	0.29	85
P20	150	3	1046.14	CL	1048.75	1042.21	1.74	83	1042.10	**1030.78**	1.10	168
P21	199	3	1337.83	CL	1281.58	1272.41	0.77	110	1283.03	**1271.75**	0.89	217
P22	120	3	1012.17	CL	1010.30	1008.78	0.16	65	1008.81	**1008.71**	0.01	130
P23	100	3	818.75	CL	807.67	**803.29**	0.55	37	807.00	**803.29**	0.46	73
PR01	48	4	1384.15	CL	1387.37	**1380.77**	0.48	10	1393.85	**1380.77**	0.95	19
PR02	96	4	2320.97	CL	2311.54	**2311.54**	0.00	32	2330.60	**2311.54**	0.82	63
PR03	144	4	2623.31	CL	2608.31	**2590.01**	0.71	71	2607.66	2602.13	0.68	140
PR04	192	4	3500.79	CL	3510.26	3481.44	1.04	98	3489.51	**3474.01**	0.45	191
PR05	240	4	4479.34	CL	4430.28	**4382.65**	1.09	123	4431.16	4416.38	1.11	251
PR06	288	4	4546.79	CL	4475.52	4452.93	0.70	159	4465.18	**4444.52**	0.47	314
PR07	72	6	1955.11	CL	1926.52	**1889.82**	1.94	19	1916.50	**1889.82**	1.41	39
PR08	144	6	3082.32	CL	3001.88	**2976.76**	0.84	66	3007.99	2977.50	1.05	135
PR09	216	6	3664.22	CL	3581.58	3548.22	1.28	113	3567.15	**3536.20**	0.88	226
PR10	288	6	4739.43	CL	4675.65	**4646.96**	0.62	162	4673.67	4648.76	0.57	322
PR11	1008	4	13227.96	CL	12987.58	12888.47	2.11	433	12810.71	**12719.65**	0.72	847
PR12	720	6	9621.99	CL	9510.37	9437.14	1.30	332	9437.56	**9388.07**	0.53	658
Tot.			90274		89169.30	88541.88		2853	88810.17	88273.77		5658
Avg.							0.80	81			0.60	162
< PB						29				30		
#B			5			18				30		

Table 10: Site dependent vehicle routing problems. The table should be interpreted like Table 9. Column *t* shows the number of vehicle types. *CL* refers to the the heuristic by Cordeau and Laporte [18] and *CGW* refers to the heuristic by Chao et al [12]. The ALNS heuristic was applied 10 times for each problem.

Heuristic	CPU	Christofides %	Christofides Minutes	Golden et al. %	Golden et al. Minutes	Li et al. %	Li et al. Minutes
TV	P-200MHz	0.64	3.84	2.88	17.55	-	-
LGV	Athlon 1Ghz	-	-	1.05	-	1.20	3.16
CGLM	P4-2GHz	0.56	24.62	1.46	56.11	-	-
EOS	P3-733MHz	0.24	30.95	3.77	137.95	-	-
P	P3-1GHz	0.24	5.19	0.92	66.90	-	-
TK	P2-400MHz	0.23	5.22	-	-	-	-
MB Best	P4-2GHz	0.03	7.72	0.01	72.94	-	-
MB Fast	P4-2GHz	0.07	0.27	0.94	0.63	-	-
BB	P-400MHz	0.49	21.25	-	-	-	-
RDH	P-900Mhz	-	-	0.67	49.33	-	-
ALNS 25K Best of 10	P4-3GHz	0.15	9.33	0.67	53.00	0.88	243.17
ALNS 25K Avg.	P4-3GHz	0.39	0.93	1.25	5.30	2.40	24.32
ALNS 50K Best of 10	P4-3GHz	0.11	17.50	0.49	107.67	0.50	497.90
ALNS 50K Avg.	P4-3GHz	0.31	1.75	1.02	10.77	1.90	49.79
		14 instances 50-200 customers		20 instances 240-483 customers		12 instances 560-1200 customers	

Table 11: Capacitated vehicle routing problems. The table compares the ALNS heuristic to nine heuristics proposed in the literature recently. The first column indicates the heuristic considered. *TV* — granular tabu search by Toth and Vigo [58], *LGV* — variable-length neighbor list record-to-record travel heuristic by Li et al. [40], *CGLM* — unified tabu search by Cordeau et al. [17, 19], *EOS* — very large scale neighborhood search by Ergun et al [23], *P* — evolutionary algorithm by Prins [46], *TK* — bone route heuristic by Tarantilis and Kiranoudis [57], *MB* — AGES heuristic by Mester and Bräysy [42] (two configurations of this heuristic is included in the table), *BB* — hybrid genetic algorithm by Berger and Barkaoui [3], *RDH* — ants system algorithm by Reimann et al. [47]. The table contains four rows for the ALNS heuristic. For each of the configurations ALNS-25K and ALNS-50K we report the best solution quality in ten experiments and the average solution quality (averaged over the same ten experiments). The *CPU* column lists the CPU used, *P* is used as an abbreviation for Pentium. The rest of the table contains three major columns, one for each dataset. For each of the datasets we report the gap between the solution obtained by the heuristic and the best known solution and we report the time spend on average by the heuristic to solve one instance. When reporting solution times for finding the best solution of ten runs, the time of all runs has been included. The ALNS heuristic is the only heuristic that has been applied to all datasets, which explains the missing entries. It should be noted that some of the numbers reported in the table were obtained from the survey by Cordeau et al. [16].

	Best known				ALNS 25K						ALNS 50K					
	n	#veh.	cost	References	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)
P01	50	**5**	**408.5**	FEL	416.67	5.0	416.06	5	2.00	12	416.45	5.0	416.06	5	1.95	23
P02	75	10	570.6	FEL	570.81	10.0	**567.14**	**10**	0.65	36	568.86	10.0	**567.14**	**10**	0.30	53
P03	100	**8**	**617**	FEL	642.93	8.0	641.76	8	4.20	85	642.32	8.0	641.76	8	4.10	128
P04	150	12	734.5	FEL	734.34	12.0	**733.13**	**12**	0.17	179	733.49	12.0	**733.13**	**12**	0.05	279
P05	199	16	953.4	B	912.54	16.0	897.93	16	1.84	124	907.03	16.0	**896.08**	**16**	1.22	237
P06	50	**6**	**400.6**	FEL	412.96	6.0	412.96	6	3.08	20	412.96	6.0	412.96	6	3.08	31
P07	75	10	634.5	B	592.16	10.0	584.15	10	1.54	18	588.72	10.0	**583.19**	**10**	0.95	33
P08	100	**9**	**638.2**	FEL	646.23	9.0	645.31	9	1.26	73	646.28	9.0	645.16	9	1.27	114
P09	150	13	785.2	B	766.42	13.1	759.35	13	1.13	108	764.32	13.1	**757.84**	**13**	0.85	185
P10	199	17	884.6	B	882.33	17.0	**875.67**	**17**	0.76	120	878.42	17.0	**875.67**	**17**	0.31	224
P11	120	7	683.4	B	682.68	7.0	**682.12**	**7**	0.08	73	682.39	7.0	**682.12**	**7**	0.04	141
P12	100	10	534.8	FEL	534.81	10.0	**534.24**	**10**	0.11	80	534.44	10.0	**534.24**	**10**	0.04	118
P13	120	11	943.7	B	911.98	11.0	**909.80**	**11**	0.24	61	911.12	11.0	**909.80**	**11**	0.15	116
P14	100	11	597.3	B	591.87	11.0	**591.87**	**11**	0.00	40	591.89	11.0	**591.87**	**11**	0.00	75
F11	71	**4**	**175**	FEL	177.00	4.0	177.00	4	1.14	69	177.00	4.0	177.00	4	1.14	104
F12	134	7	778.5	FEL	770.59	7.0	**770.17**	**7**	0.06	237	770.31	7.0	**770.17**	**7**	0.02	359
Tot.		156	10340		10246.32	156.10	10198.67	156		1336	10225.99	156.10	10194.19	156		2222
Avg.									1.14	83					0.97	139
< PB					11						11					
#B	5				8						11					

Table 12: Open vehicle routing problem instances. The table should be interpreted like Table 9. The abbreviations used in the *References* column are: *B* - Brandao's heuristic [6], *FEL* - the heuristic by Fu et al. [25]. The column *#veh.* indicates the number of vehicles used in the previous best solution, *avg. #veh.* indicates the number of vehicles used on average by the particular ALNS configuration (averaged over ten experiments). The column *best #veh.* indicates the number of vehicles used in the best found solution (out of 10 experiments).

	Best known			ALNS 25K						ALNS 50K					
	veh.	cost	References	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)
R101	**19**	**1645.79**	H (2000)	1650.86	19.0	1650.80	19	0.31	55	1650.80	19.0	1650.80	19	0.30	85
R102	**17**	**1486.12**	RT (1995)	1486.89	17.0	**1486.12**	**17**	0.05	62	1486.75	17.0	**1486.12**	**17**	0.04	94
R103	**13**	**1292.68**	LLH (2001)	1294.89	13.0	**1292.68**	**13**	0.17	64	1294.04	13.0	**1292.68**	**13**	0.11	97
R104	**9**	**1007.24**	M (2002)	987.85	9.8	1013.13	9	-1.93	61	987.85	9.8	1013.13	9	-1.93	96
R105	**14**	**1377.11**	RT (1995)	1378.77	14.0	**1377.11**	**14**	0.12	56	1378.11	14.0	**1377.11**	**14**	0.07	85
R106	**12**	**1251.98**	M (2002)	1258.40	12.0	1252.03	12	0.51	61	1255.52	12.0	1252.03	12	0.28	92
R107	**10**	**1104.55**	S97 (1997)	1118.18	10.0	1113.70	10	1.23	52	1115.19	10.0	1104.76	10	0.96	85
R108	**9**	**960.88**	BBB (2001)	969.37	9.0	963.91	9	0.88	40	965.36	9.0	**960.88**	**9**	0.47	75
R109	**11**	**1194.73**	HG (1999)	1213.09	11.1	**1194.73**	**11**	1.54	47	1211.44	11.1	**1194.73**	**11**	1.40	77
R110	**10**	**1118.59**	M (2002)	1149.56	10.0	1119.14	10	2.77	41	1148.92	10.0	1119.14	10	2.71	71
R111	**10**	**1096.72**	RGP (2001?)	1112.14	10.0	1096.74	10	1.41	46	1105.36	10.0	1096.73	10	0.79	78
R112	**9**	**982.14**	GTA (1999)	983.16	9.5	1000.60	9	0.10	58	982.62	9.5	1000.60	9	0.05	91
C101	**10**	**828.94**	RT (1995)	828.94	10.0	**828.94**	**10**	0.00	29	828.94	10.0	**828.94**	**10**	0.00	57
C102	**10**	**828.94**	RT (1995)	828.94	10.0	**828.94**	**10**	0.00	59	828.94	10.0	**828.94**	**10**	0.00	91
C103	**10**	**828.06**	RT (1995)	828.06	10.0	**828.06**	**10**	0.00	65	828.06	10.0	**828.06**	**10**	0.00	99
C104	**10**	**824.78**	RT (1995)	824.78	10.0	**824.78**	**10**	0.00	69	824.78	10.0	**824.78**	**10**	0.00	105
C105	**10**	**828.94**	RT (1995)	828.94	10.0	**828.94**	**10**	0.00	31	828.94	10.0	**828.94**	**10**	0.00	59
C106	**10**	**828.94**	RT (1995)	828.94	10.0	**828.94**	**10**	0.00	32	828.94	10.0	**828.94**	**10**	0.00	62
C107	**10**	**828.94**	RT (1995)	828.94	10.0	**828.94**	**10**	0.00	32	828.94	10.0	**828.94**	**10**	0.00	62
C108	**10**	**828.94**	RT (1995)	828.94	10.0	**828.94**	**10**	0.00	61	828.94	10.0	**828.94**	**10**	0.00	93
C109	**10**	**828.94**	RT (1995)	828.94	10.0	**828.94**	**10**	0.00	64	828.94	10.0	**828.94**	**10**	0.00	99
RC101	**14**	**1696.94**	TBGGP (1997)	1688.35	14.2	1697.43	14	-0.51	53	1688.17	14.2	1697.43	14	-0.52	80
RC102	**12**	**1554.75**	TBGGP (1997)	1547.04	12.1	**1554.75**	**12**	-0.50	56	1555.06	12.1	**1554.75**	**12**	0.02	84
RC103	**11**	**1261.67**	S98 (1998)	1270.78	11.0	1262.02	11	0.72	58	1268.53	11.0	1262.02	11	0.54	90
RC104	**10**	**1135.48**	CLM (2000)	1135.80	10.0	1135.52	10	0.03	60	1135.89	10.0	1135.83	10	0.04	92
RC105	**13**	**1629.44**	BBB (2001)	1640.18	13.0	**1629.44**	**13**	0.66	54	1640.92	13.0	1633.72	13	0.70	83
RC106	**11**	**1424.73**	BBB (2001)	1413.07	11.5	1432.12	11	-0.82	49	1411.92	11.5	1432.12	11	-0.90	76
RC107	**11**	**1230.48**	S97 (1997)	1232.48	11.0	1230.95	11	0.16	56	1231.65	11.0	1230.54	11	0.09	86
RC108	**10**	**1139.82**	TBGGP (1997)	1167.55	10.0	1140.87	10	2.43	41	1152.30	10.0	**1139.82**	**10**	1.10	71
R201	**4**	**1252.37**	HG (1999)	1253.23	4.0	1253.23	4	0.07	133	1253.23	4.0	1253.23	4	0.07	193
R202	**3**	**1191.7**	RGP (2001?)	1229.81	3.0	1195.30	3	3.20	96	1223.62	3.0	1195.30	3	2.68	181
R203	**3**	**939.54**	M (2002)	944.64	3.0	939.58	3	0.54	164	943.57	3.0	941.08	3	0.43	256
R204	**2**	**825.52**	BVH (2001)	841.48	2.0	833.09	2	1.93	182	843.39	2.0	833.09	2	2.16	346
R205	**3**	**994.42**	RGP (2001?)	1018.90	3.0	994.43	3	2.46	97	1010.43	3.0	994.43	3	1.61	186
R206	**3**	**906.14**	SSSD (2000)	923.91	3.0	915.27	3	1.96	192	921.07	3.0	**906.14**	**3**	1.65	282
R207	**2**	**893.33**	BVH (2001)	928.28	2.0	**893.33**	**2**	3.91	180	927.62	2.0	**893.33**	**2**	3.84	332
R208	**2**	**726.75**	M (2002)	736.12	2.0	726.82	2	1.29	185	735.76	2.0	726.82	2	1.24	369
R209	**3**	**909.16**	H (2000)	926.72	3.0	914.45	3	1.93	101	923.48	3.0	914.13	3	1.58	185
R210	**3**	**939.34**	M (2002)	955.02	3.0	954.12	3	1.67	112	955.29	3.0	950.52	3	1.70	204
R211	**2**	**892.71**	BVH (2001)	889.99	2.3	925.03	2	-0.30	216	887.93	2.3	926.83	2	-0.54	349
C201	**3**	**591.56**	RT (1995)	591.56	3.0	**591.56**	**3**	0.00	78	591.56	3.0	**591.56**	**3**	0.00	147
C202	**3**	**591.56**	RT (1995)	591.56	3.0	**591.56**	**3**	0.00	88	591.56	3.0	**591.56**	**3**	0.00	163
C203	**3**	**591.17**	RT (1995)	591.17	3.0	**591.17**	**3**	0.00	96	591.17	3.0	**591.17**	**3**	0.00	181
C204	**3**	**590.6**	RT (1995)	590.60	3.0	**590.60**	**3**	0.00	102	590.60	3.0	**590.60**	**3**	0.00	189
C205	**3**	**588.88**	RT (1995)	588.88	3.0	**588.88**	**3**	0.00	81	588.88	3.0	**588.88**	**3**	0.00	155
C206	**3**	**588.49**	RT (1995)	588.49	3.0	**588.49**	**3**	0.00	83	588.49	3.0	**588.49**	**3**	0.00	156
C207	**3**	**588.29**	RT (1995)	588.29	3.0	**588.29**	**3**	0.00	84	588.29	3.0	**588.29**	**3**	0.00	167
C208	**3**	**588.32**	RT (1995)	588.32	3.0	**588.32**	**3**	0.00	85	588.32	3.0	**588.32**	**3**	0.00	161
RC201	**4**	**1406.91**	M (2002)	1417.80	4.0	1413.52	4	0.77	83	1414.69	4.0	1413.52	4	0.55	140
RC202	**3**	**1367.09**	CC (2002)	1405.16	3.0	1368.04	3	2.78	96	1403.60	3.0	**1367.09**	**3**	2.67	177
RC203	**3**	**1049.62**	CC (2002)	1075.51	3.0	1068.08	3	2.47	100	1072.57	3.0	1068.60	3	2.19	192
RC204	**3**	**798.41**	M (2002)	818.00	3.0	799.27	3	2.45	228	806.81	3.0	798.46	3	1.05	320
RC205	**4**	**1297.19**	M (2002)	1318.01	4.0	1302.42	4	1.60	134	1312.75	4.0	1302.42	4	1.20	194
RC206	**3**	**1146.32**	H (2000)	1155.91	3.0	**1146.32**	**3**	0.84	87	1155.16	3.0	**1146.32**	**3**	0.77	166
RC207	**3**	**1061.14**	BVH (2001)	1095.29	3.0	1070.85	3	3.22	96	1088.15	3.0	1061.84	3	2.55	182
RC208	**3**	**828.14**	IKMUY (2001)	834.83	3.0	829.69	3	0.81	109	829.96	3.0	829.69	3	0.22	196
Tot.	405	57192		57641.28	407.50	57360.86	405		4800	57549.75	407.50	57332.03	405		8182
Avg.								0.77	86					0.61	146
< PB				0						0					
#B	56			25						28					

Table 13: Solomon VRPTW instances. The table should be interpreted as table 12. The best known solutions were gathered from the web page: http://www.sintef.no/static/am/opti/projects/top/vrp/benchmarks.html. See this page for complete references to where the best known solutions first were identified.

	Best known			ALNS 25K						ALNS 50K					
	veh.	cost	References	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)
R1_2_1	**19**	**5024.65**	B	4809.44	20.0	4798.22	20	-4.28	170	4798.77	20.0	4785.96	20	-4.50	257
R1_2_2	**18**	**4054.44**	MB	4091.46	18.0	4066.91	18	0.91	165	4079.18	18.0	4059.57	18	0.61	256
R1_2_3	**18**	**3164.41**	LC	3414.89	18.0	3387.64	18	7.92	160	3407.48	18.0	3396.47	18	7.68	260
R1_2_4	**18**	**3067.93**	MB	3104.90	18.0	3086.11	18	1.20	200	3100.94	18.0	3086.65	18	1.08	308
R1_2_5	**18**	**4112.88**	MB	4184.89	18.0	4125.19	18	1.75	147	4157.28	18.0	4125.19	18	1.08	231
R1_2_6	18	3599.84	MB	3643.10	18.0	3616.52	18	1.57	167	3631.74	18.0	**3586.80**	**18**	1.25	258
R1_2_7	**18**	**3151.42**	MB	3187.56	18.0	3170.98	18	1.15	169	3186.04	18.0	3160.44	18	1.10	271
R1_2_8	**18**	**2963.9**	MB	2992.96	18.0	2971.66	18	0.98	200	2989.62	18.0	2975.59	18	0.87	310
R1_2_9	**18**	**3784.33**	MB	3853.46	18.0	3802.55	18	1.83	135	3840.07	18.0	3823.15	18	1.47	223
R1_210	**18**	**3307.78**	MB	3363.82	18.0	3333.66	18	1.69	148	3336.35	18.0	3312.44	18	0.86	241
C1_2_1	**20**	**2704.57**	GH	2704.57	20.0	**2704.57**	**20**	0.00	94	2704.57	20.0	**2704.57**	**20**	0.00	181
C1_2_2	**18**	**2917.89**	BVH	2977.48	18.0	2948.73	18	2.04	152	2969.62	18.0	2943.83	18	1.77	242
C1_2_3	**18**	**2708.08**	MB	2744.41	18.0	2719.62	18	1.34	145	2729.39	18.0	2710.21	18	0.79	245
C1_2_4	**18**	**2644.61**	MB	2646.94	18.0	2645.60	18	0.09	146	2646.36	18.0	2644.92	18	0.07	253
C1_2_5	**20**	**2702.05**	GH	2702.05	20.0	**2702.05**	**20**	0.00	96	2702.05	20.0	**2702.05**	**20**	0.00	186
C1_2_6	**20**	**2701.04**	GH	2701.04	20.0	**2701.04**	**20**	0.00	101	2701.04	20.0	**2701.04**	**20**	0.00	193
C1_2_7	**20**	**2701.04**	GH	2701.04	20.0	**2701.04**	**20**	0.00	184	2701.04	20.0	**2701.04**	**20**	0.00	281
C1_2_8	**18**	**2769.19**	MB	2791.15	19.0	2775.48	19	0.79	157	2789.38	19.0	2775.48	19	0.73	250
C1_2_9	**18**	**2642.82**	MB	2705.26	18.0	2687.83	18	2.36	104	2688.82	18.0	2687.83	18	1.74	196
C1_210	18	2649.26	MB	2650.64	18.0	2645.08	18	0.24	117	2651.55	18.0	**2644.25**	**18**	0.28	214
RC1_2_1	18	3691.99	MB	3812.41	18.0	3727.17	18	4.52	93	3731.52	18.0	**3647.56**	**18**	2.30	175
RC1_2_2	18	3298.68	MB	3342.07	18.0	**3269.91**	**18**	2.21	96	3309.57	18.0	3276.88	18	1.21	185
RC1_2_3	**18**	**3025.9**	MB	3053.11	18.0	3036.32	18	0.90	104	3051.91	18.0	3034.45	18	0.86	201
RC1_2_4	18	2879.4	MB	2906.27	18.0	**2869.74**	**18**	1.27	109	2887.58	18.0	2873.54	18	0.62	215
RC1_2_5	**18**	**3419.81**	MB	3509.40	18.0	3463.01	18	2.62	90	3500.46	18.0	3430.03	18	2.36	173
RC1_2_6	18	3393.09	MB	3473.96	18.0	3398.67	18	3.46	91	3431.75	18.0	**3357.90**	**18**	2.20	174
RC1_2_7	18	3266.48	MB	3353.23	18.0	3290.65	18	3.71	93	3302.54	18.0	**3233.29**	**18**	2.14	179
RC1_2_8	18	3115.82	MB	3163.78	18.0	3147.87	18	1.71	95	3149.37	18.0	**3110.46**	**18**	1.25	183
RC1_2_9	**18**	**3083.41**	MB	3152.09	18.0	3114.02	18	2.23	94	3150.15	18.0	3116.47	18	2.16	183
RC1_210	18	3038.85	MB	3063.57	18.0	**3020.24**	**18**	1.43	98	3056.83	18.0	3042.24	18	1.21	190
R2_2_1	**4**	**4501.8**	MB	4340.82	4.5	4563.55	4	-3.58	527	4329.15	4.5	4571.67	4	-3.84	821
R2_2_2	**4**	**3645.38**	MB	3683.64	4.0	3666.72	4	1.05	416	3669.25	4.0	3650.54	4	0.65	795
R2_2_3	4	2932.44	MB	2928.17	4.0	**2892.07**	**4**	1.25	458	2924.73	4.0	**2892.07**	**4**	1.13	890
R2_2_4	**4**	**1981.29**	MB	1992.90	4.0	1981.30	4	0.59	482	1989.24	4.0	1981.30	4	0.40	910
R2_2_5	**4**	**3367.55**	SAM::OPT	3431.26	4.0	3382.22	4	1.89	385	3417.75	4.0	3377.18	4	1.49	723
R2_2_6	**4**	**2914.56**	MB	2957.14	4.0	2929.72	4	1.46	414	2947.20	4.0	2931.14	4	1.12	807
R2_2_7	**4**	**2453.62**	MB	2461.82	4.0	2456.71	4	0.33	461	2465.00	4.0	2459.82	4	0.46	883
R2_2_8	**4**	**1849.87**	MB	1874.00	4.0	1850.85	4	1.30	495	1866.03	4.0	**1849.87**	**4**	0.87	959
R2_2_9	**4**	**3111.41**	MB	3134.41	4.0	3113.74	4	0.74	405	3126.66	4.0	3113.74	4	0.49	768
R2_210	**4**	**2657**	MB	2696.24	4.0	2666.10	4	1.48	399	2690.93	4.0	2666.35	4	1.28	784
C2_2_1	**6**	**1931.44**	GH	1931.44	6.0	**1931.44**	**6**	0.00	214	1931.44	6.0	**1931.44**	**6**	0.00	391
C2_2_2	**6**	**1863.16**	GH	1863.16	6.0	**1863.16**	**6**	0.00	233	1863.16	6.0	**1863.16**	**6**	0.00	445
C2_2_3	**6**	**1775.11**	M	1784.79	6.0	1776.96	6	0.55	263	1783.42	6.0	1776.96	6	0.47	497
C2_2_4	6	1720.09	MB	1719.58	6.0	**1713.46**	**6**	0.36	275	1715.66	6.0	**1713.46**	**6**	0.13	527
C2_2_5	**6**	**1878.85**	BVH	1881.87	6.0	1879.31	6	0.16	224	1879.27	6.0	**1878.85**	**6**	0.02	413
C2_2_6	**6**	**1857.35**	B	1859.74	6.0	**1857.35**	**6**	0.13	224	1857.35	6.0	**1857.35**	**6**	0.00	425
C2_2_7	**6**	**1849.46**	GH	1851.62	6.0	**1849.46**	**6**	0.12	231	1849.46	6.0	**1849.46**	**6**	0.00	431
C2_2_8	6	1820.59	MB	1828.56	6.0	1823.88	6	0.44	233	1823.21	6.0	**1820.53**	**6**	0.15	442
C2_2_9	6	1830.18	SAM::OPT	1833.78	6.0	**1830.05**	**6**	0.20	241	1834.31	6.0	**1830.05**	**6**	0.23	449
C2_210	**6**	**1806.6**	M	1809.46	6.0	1808.21	6	0.16	249	1809.47	6.0	1808.21	6	0.16	466
RC2_2_1	**6**	**3103.48**	MB	3146.36	6.0	3126.03	6	1.38	428	3143.65	6.0	3129.07	6	1.29	635
RC2_2_2	**5**	**2827.45**	M	2870.43	5.0	2828.39	5	1.52	629	2856.08	5.0	2835.67	5	1.01	916
RC2_2_3	4	2617.9	MB	2652.00	4.0	2620.87	4	1.49	444	2631.97	4.0	**2613.12**	**4**	0.72	849
RC2_2_4	4	2055.97	MB	2080.99	4.0	2056.93	4	1.38	476	2063.32	4.0	**2052.74**	**4**	0.52	923
RC2_2_5	4	2912.57	MB	3039.09	4.0	2913.21	4	4.36	433	3041.01	4.0	**2912.13**	**4**	4.43	803
RC2_2_6	4	3086.76	LC	2920.37	4.3	2977.41	4	-1.84	525	2900.85	4.3	**2975.13**	**4**	-2.50	846
RC2_2_7	4	2550.56	M	2609.93	4.0	2563.90	4	2.76	408	2572.96	4.0	**2539.85**	**4**	1.30	789
RC2_2_8	4	2317.8	MB	2341.46	4.0	2322.52	4	1.16	413	2330.66	4.0	**2314.61**	**4**	0.69	788
RC2_2_9	**4**	**2175.61**	MB	2216.32	4.0	2175.98	4	1.87	417	2214.61	4.0	2180.81	4	1.79	795
RC2_210	**4**	**2015.6**	MB	2046.95	4.0	2020.68	4	1.56	424	2030.64	4.0	2015.61	4	0.75	808
Tot.	692	168997		170589.23	694.80	169370.28	694		15341	169941.42	694.80	169042.17	694		27688
Avg.								1.17	256					0.81	461
< PB				8						18					
#B	41			14						26					

Table 14: Gehring/Homberger VRPTW instances, 200 customers.. The best known solutions were gathered from the web page: http://www.sintef.no/static/am/opti/projects/top/vrp/benchmarks.html in January 2005. This list of best known solutions was supplemented by the solutions found by Mester and Bräysy [42] (MB) and Le Bouthillier and Crainic [5] (LC). See the aforementioned web page for full references. The same sources were used for the best known solution columns in tables 4 to 7.

	Best known			ALNS 25K						ALNS 50K					
	veh.	cost	References	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)
R1_4_1	**38**	**11084**	B	10557.71	40.0	10502.22	40	-4.75	368	10485.89	40.0	10432.30	40	-5.40	554
R1_4_2	36	9161.26	MB	9277.12	36.0	9239.87	36	1.77	224	9166.43	36.0	**9115.68**	**36**	0.56	401
R1_4_3	**36**	**7941.53**	MB	8029.69	36.0	7996.33	36	1.11	269	8053.08	36.0	7988.22	36	1.40	464
R1_4_4	**36**	**7332.93**	MB	7468.64	36.0	7449.60	36	1.85	227	7441.43	36.0	7415.81	36	1.48	439
R1_4_5	36	9512.25	MB	9738.05	36.0	9588.45	36	2.73	182	9560.46	36.0	**9479.10**	**36**	0.86	347
R1_4_6	**36**	**8534.05**	MB	8740.69	36.0	8677.13	36	2.42	227	8613.60	36.0	8556.38	36	0.93	407
R1_4_7	**36**	**7710.41**	MB	7812.04	36.0	7769.68	36	1.32	245	7763.04	36.0	7725.97	36	0.68	439
R1_4_8	36	7398.68	MB	7468.69	36.0	7425.43	36	1.05	231	7398.80	36.0	**7390.76**	**36**	0.11	444
R1_4_9	**36**	**8878.19**	MB	9125.58	36.0	9058.30	36	2.79	217	9053.20	36.0	8970.98	36	1.97	386
R1_410	**36**	**8227.49**	MB	8417.50	36.0	8386.75	36	2.31	194	8363.10	36.0	8325.16	36	1.65	377
C1_4_1	**40**	**7152.02**	M	7152.06	40.0	7152.06	40	0.00	203	7152.06	40.0	7152.06	40	0.00	387
C1_4_2	37	7357.45	MB	7815.71	36.2	7830.99	36	1.06	260	7759.63	36.2	**7733.55**	**36**	0.34	437
C1_4_3	36	7151.17	MB	7208.25	36.0	7174.23	36	1.78	212	7104.35	36.0	**7082.13**	**36**	0.31	403
C1_4_4	36	6822.18	MB	6909.71	36.0	6833.32	36	1.37	224	6861.13	36.0	**6816.17**	**36**	0.66	431
C1_4_5	**40**	**7152.02**	M	7152.06	40.0	7152.06	40	0.00	215	7152.06	40.0	7152.06	40	0.00	404
C1_4_6	**40**	**7153.41**	M	7153.45	40.0	7153.45	40	0.00	286	7153.45	40.0	7153.45	40	0.00	479
C1_4_7	39	7668.33	LC	7643.60	39.0	7620.09	39	1.28	297	7621.62	39.0	**7546.78**	**39**	0.99	485
C1_4_8	38	7113.4	MB	7814.18	37.0	7661.98	37	3.55	284	7794.27	37.0	**7546.32**	**37**	3.29	464
C1_4_9	**36**	**7524.32**	MB	8042.29	36.0	7673.65	36	6.88	255	7800.59	36.0	7573.18	36	3.67	428
C1_410	**36**	**6907.26**	MB	7617.12	36.0	7446.94	36	10.28	214	7325.70	36.0	7145.92	36	6.06	398
RC1_4_1	36	8960.82	MB	9139.22	36.0	9044.65	36	3.70	207	8939.82	36.0	**8813.43**	**36**	1.43	371
RC1_4_2	36	8174.27	MB	8287.21	36.0	8181.05	36	2.08	197	8176.96	36.0	**8118.43**	**36**	0.72	370
RC1_4_3	36	7737.99	MB	7744.57	36.0	7668.27	36	1.05	214	7729.95	36.0	**7663.73**	**36**	0.86	403
RC1_4_4	36	7411.02	MB	7497.41	36.0	7447.70	36	1.75	226	7433.65	36.0	**7368.47**	**36**	0.88	436
RC1_4_5	36	8499.15	MB	8634.51	36.0	8503.19	36	2.47	190	8520.69	36.0	**8426.57**	**36**	1.12	356
RC1_4_6	**36**	**8304.99**	MB	8640.29	36.0	8533.72	36	4.04	185	8445.05	36.0	8390.24	36	1.69	351
RC1_4_7	**36**	**8051.71**	MB	8355.82	36.0	8223.65	36	3.78	192	8331.40	36.0	8227.10	36	3.47	360
RC1_4_8	**36**	**7917.68**	MB	8174.94	36.0	8135.05	36	3.25	192	8070.47	36.0	7922.67	36	1.93	363
RC1_4_9	**36**	**7890.45**	MB	8067.40	36.0	7953.20	36	2.24	194	8016.28	36.0	7987.55	36	1.59	370
RC1_410	**36**	**7716.32**	MB	7861.40	36.0	7805.59	36	1.88	199	7823.83	36.0	7774.83	36	1.39	376
R2_4_1	**8**	**9257.92**	MB	9513.88	8.0	9375.10	8	2.76	1002	9432.87	8.0	9338.49	8	1.89	1574
R2_4_2	8	7674.9	MB	7762.67	8.0	7728.27	8	1.47	1313	7744.54	8.0	**7649.87**	**8**	1.24	1942
R2_4_3	**8**	**5988.02**	MB	6078.27	8.0	5998.04	8	1.51	1426	6053.22	8.0	6034.08	8	1.09	2120
R2_4_4	8	4331.07	MB	4356.73	8.0	**4326.48**	**8**	0.70	1565	4345.23	8.0	4327.61	8	0.43	2333
R2_4_5	**8**	**7143.55**	MB	7305.24	8.0	7255.52	8	2.26	1207	7277.89	8.0	7252.64	8	1.88	1841
R2_4_6	**8**	**6163.81**	MB	6284.34	8.0	6222.32	8	1.96	1326	6229.61	8.0	6212.37	8	1.07	1986
R2_4_7	**8**	**5082.1**	MB	5182.15	8.0	5138.58	8	1.97	1441	5154.64	8.0	5136.74	8	1.43	2164
R2_4_8	8	4068.97	MB	4090.90	8.0	**4055.22**	**8**	0.88	1587	4076.34	8.0	4060.51	8	0.52	2384
R2_4_9	**8**	**6493.13**	MB	6565.87	8.0	6526.20	8	1.12	1222	6537.26	8.0	6507.40	8	0.68	1817
R2_410	8	5895.93	MB	5958.31	8.0	**5894.40**	**8**	1.08	1283	5919.14	8.0	5897.46	8	0.42	1891
C2_4_1	**12**	**4116.05**	M	4125.50	12.0	4116.33	12	0.23	403	4116.93	12.0	4116.33	12	0.02	753
C2_4_2	12	3930.29	MB	3930.22	12.0	**3930.05**	**12**	0.00	477	3930.13	12.0	**3930.05**	**12**	0.00	858
C2_4_3	**12**	**3739.72**	GH	3782.86	12.0	3775.32	12	1.15	525	3780.81	12.0	3775.54	12	1.10	952
C2_4_4	**12**	**3535.99**	MB	3549.80	12.0	3546.66	12	0.39	520	3568.37	12.0	3543.60	12	0.92	965
C2_4_5	**12**	**3939.42**	MB	3981.35	12.0	3946.94	12	1.06	434	3951.72	12.0	3946.14	12	0.31	783
C2_4_6	**12**	**3875.94**	MB	3883.95	12.0	**3875.94**	**12**	0.21	457	3921.04	12.0	**3875.94**	**12**	1.16	811
C2_4_7	**12**	**3894.13**	M	3937.44	12.0	3903.46	12	1.11	453	3960.36	12.0	3894.98	12	1.70	829
C2_4_8	**12**	**3787.08**	MB	3863.49	12.0	3804.12	12	2.02	498	3850.01	12.0	3796.00	12	1.66	884
C2_4_9	**12**	**3876.1**	MB	4025.46	12.0	3887.00	12	3.85	471	3964.79	12.0	3881.21	12	2.29	851
C2_410	**12**	**3684.89**	MB	3764.34	12.0	3706.87	12	2.16	502	3715.36	12.0	3687.13	12	0.83	896
RC2_4_1	11	7019.89	GH	6876.33	11.2	**6834.02**	**11**	0.62	786	6857.62	11.2	6840.51	11	0.35	1198
RC2_4_2	10	5924.84	MB	6166.44	9.8	6356.23	9	-2.98	1029	6125.49	9.8	**6355.59**	**9**	-3.62	1553
RC2_4_3	8	5114.76	MB	5139.79	8.0	5073.80	8	1.68	820	5109.29	8.0	**5055.02**	**8**	1.07	1503
RC2_4_4	8	3648.64	MB	3737.66	8.0	3666.70	8	2.47	959	3692.45	8.0	**3647.39**	**8**	1.24	1694
RC2_4_5	**9**	**6063.46**	MB	6107.39	9.4	6257.87	9	0.72	901	6019.04	9.4	6119.44	9	-0.73	1416
RC2_4_6	8	6054.21	GH	6093.66	8.0	**5997.24**	**8**	1.61	835	6092.17	8.0	6008.41	8	1.58	1425
RC2_4_7	8	5519.25	MB	5664.90	8.0	5529.42	8	3.44	714	5623.09	8.0	**5476.57**	**8**	2.68	1324
RC2_4_8	**8**	**4854.16**	MB	4949.32	8.0	4877.39	8	1.96	1284	4933.15	8.0	4891.18	8	1.63	1903
RC2_4_9	8	4628.26	MB	4736.64	8.0	4674.88	8	2.94	1168	4662.33	8.0	**4601.30**	**8**	1.33	1779
RC2_410	**8**	**4316.36**	MB	4415.46	8.0	4400.68	8	2.30	1290	4401.00	8.0	4355.52	8	1.96	1937
Tot.	1386	392070		399377.24	1386.60	395969.66	1385		34732	396157.93	1386.60	393210.00	1385		56699
Avg.								1.80	579					1.05	945
< PB						12						24			
#B		35				7						21			

Table 15: Gehring/Homberger VRPTW instances, 400 customers..

	Best known			ALNS 25K						ALNS 50K					
	veh.	cost	References	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)
R1_6_1	**59**	**21131.09**	MB	21881.08	59.0	21767.25	59	3.55	514	21743.91	59.0	21677.41	59	2.90	763
R1_6_2	**54**	**19603.7**	MB	20892.38	54.0	20719.50	54	6.57	276	20253.42	54.0	20045.49	54	3.31	504
R1_6_3	**54**	**17400.6**	MB	18399.70	54.0	18154.60	54	5.74	289	17886.87	54.0	17733.91	54	2.79	535
R1_6_4	**54**	**15993.8**	MB	16640.46	54.0	16550.00	54	4.04	300	16459.06	54.0	16374.29	54	2.91	569
R1_6_5	**54**	**20395**	MB	22399.39	54.0	22051.85	54	9.83	359	21462.92	54.0	21243.24	54	5.24	577
R1_6_6	**54**	**18620.26**	MB	19759.95	54.0	19610.14	54	6.12	263	19206.68	54.0	18948.53	54	3.15	494
R1_6_7	**54**	**17107.91**	MB	17915.15	54.0	17773.37	54	4.72	279	17483.82	54.0	17438.28	54	2.20	527
R1_6_8	**54**	**15725.86**	MB	16509.06	54.0	16436.50	54	4.98	295	16245.90	54.0	16146.17	54	3.31	560
R1_6_9	**54**	**19372.96**	MB	21316.90	54.0	20860.58	54	10.03	276	20548.47	54.0	20375.70	54	6.07	494
R1_610	**54**	**18235.57**	MB	19909.33	54.0	19776.64	54	9.18	258	19193.80	54.0	18902.19	54	5.25	485
R2_6_1	**11**	**18325.6**	MB	19066.50	11.0	18865.57	11	4.04	872	18937.51	11.0	18837.28	11	3.34	1622
R2_6_2	11	15346.42	MB	15318.18	11.0	15222.07	11	1.65	928	15187.30	11.0	**15069.24**	**11**	0.78	1727
R2_6_3	11	11663.06	MB	11422.68	11.0	11395.17	11	1.16	1001	11386.17	11.0	**11291.52**	**11**	0.84	1903
R2_6_4	11	8386.64	MB	8331.34	11.0	8264.60	11	2.06	1115	8251.65	11.0	**8163.24**	**11**	1.08	2021
R2_6_5	11	15640.6	MB	15637.54	11.0	15430.80	11	1.42	862	15558.66	11.0	**15418.00**	**11**	0.91	1621
R2_6_6	11	12937.47	MB	13133.25	11.0	13038.58	11	1.52	920	13026.65	11.0	**12936.28**	**11**	0.70	1766
R2_6_7	11	10536.84	MB	10487.56	11.0	10437.39	11	2.12	1000	10352.03	11.0	**10269.96**	**11**	0.80	1904
R2_6_8	11	8023.64	MB	7886.24	11.0	7849.32	11	1.72	1095	7805.77	11.0	**7752.78**	**11**	0.68	2086
R2_6_9	**11**	**13567.84**	MB	14181.45	11.0	14016.38	11	4.52	876	14000.78	11.0	13885.52	11	3.19	1627
R2_610	11	12607.09	MB	12799.15	11.0	12775.18	11	1.83	881	12706.72	11.0	**12568.79**	**11**	1.10	1690
C1_6_1	**60**	**14095.64**	GH	14095.64	60.0	**14095.64**	**60**	0.00	286	14095.64	60.0	**14095.64**	**60**	0.00	540
C1_6_2	56	14325.96	MB	14446.21	56.0	14179.06	56	1.92	495	14278.31	56.0	**14174.12**	**56**	0.74	737
C1_6_3	56	13898.99	MB	13866.54	56.0	13842.83	56	0.46	509	13842.21	56.0	**13803.50**	**56**	0.28	767
C1_6_4	56	13610.66	MB	13626.16	56.0	13615.92	56	0.35	538	13603.40	56.0	**13578.66**	**56**	0.18	812
C1_6_5	**60**	**14085.7**	BVH	14085.72	60.0	14085.72	60	0.00	306	14085.72	60.0	14085.72	60	0.00	564
C1_6_6	60	14089.7	BVH	14089.66	60.0	**14089.66**	**60**	0.00	413	14089.66	60.0	**14089.66**	**60**	0.00	674
C1_6_7	59	14659.74	GH	14832.65	58.6	**15017.03**	**58**	-1.23	473	14803.08	58.6	15032.51	58	-1.42	726
C1_6_8	57	14976.88	GH	14690.74	57.0	14409.78	57	2.42	433	14510.17	57.0	**14343.05**	**57**	1.17	675
C1_6_9	**56**	**13733.56**	MB	14265.06	56.0	14017.73	56	3.87	483	13883.26	56.0	13767.45	56	1.09	723
C1_610	56	13758.19	MB	14128.71	56.0	13906.05	56	3.22	492	13788.90	56.0	**13688.57**	**56**	0.73	742
C2_6_1	**18**	**7774.1**	MB	7789.40	18.0	7780.84	18	0.20	553	7791.82	18.0	7786.86	18	0.23	987
C2_6_2	18	7486.88	MB	7764.29	17.8	8800.94	17	-11.76	727	7763.97	17.8	**8799.38**	**17**	-11.77	1224
C2_6_3	17	8371.07	GH	7676.89	17.6	7795.66	17	0.96	762	7613.00	17.6	**7604.00**	**17**	0.12	1275
C2_6_4	17	7216.45	MB	7269.90	17.2	7054.65	17	3.95	722	7088.64	17.2	**6993.77**	**17**	1.36	1266
C2_6_5	**18**	**7576.35**	MB	7694.89	18.0	7592.79	18	1.56	581	7606.34	18.0	7578.12	18	0.40	1007
C2_6_6	**18**	**7478.63**	MB	8515.65	18.0	7984.40	18	13.87	635	7910.69	18.0	7554.61	18	5.78	1088
C2_6_7	18	7560.53	MB	8474.41	18.0	**7520.34**	**18**	12.69	727	8234.69	18.0	7610.04	18	9.50	1190
C2_6_8	18	7352.42	MB	7771.07	17.8	**8696.15**	**17**	-10.64	672	7734.91	17.8	8782.31	17	-11.05	1159
C2_6_9	**18**	**7350.94**	MB	7609.44	18.0	7356.19	18	3.52	669	7384.14	18.0	7364.93	18	0.45	1148
C2_610	**17**	**7523.34**	MB	7781.30	17.6	8334.99	17	3.43	656	7697.89	17.6	7938.94	17	2.32	1136
RC1_6_1	**55**	**17454.39**	MB	18210.19	55.0	17987.59	55	4.33	275	17928.76	55.0	17751.33	55	2.72	494
RC1_6_2	**55**	**16208.24**	MB	16883.37	55.0	16718.63	55	4.17	436	16686.63	55.0	16548.43	55	2.95	671
RC1_6_3	55	15524.33	MB	15968.19	55.0	15907.78	55	3.03	333	15642.26	55.0	**15499.02**	**55**	0.92	584
RC1_6_4	55	15180.72	MB	15295.11	55.0	15214.81	55	1.47	358	15192.70	55.0	**15072.90**	**55**	0.79	621
RC1_6_5	55	17468.57	MB	17981.80	55.0	17879.49	55	3.34	329	17543.75	55.0	**17401.34**	**55**	0.82	551
RC1_6_6	**55**	**17248.87**	MB	17913.64	55.0	17646.26	55	3.85	329	17466.21	55.0	17355.10	55	1.26	548
RC1_6_7	**55**	**16454.79**	MB	17484.20	55.0	17159.31	55	6.26	410	17143.21	55.0	17058.40	55	4.18	636
RC1_6_8	**55**	**16462.49**	MB	17043.31	55.0	16955.52	55	3.53	336	16705.09	55.0	16510.65	55	1.47	568
RC1_6_9	**55**	**16153**	MB	16806.32	55.0	16609.24	55	4.04	378	16525.18	55.0	16435.71	55	2.30	604
RC1_610	**55**	**16030.86**	MB	16483.29	55.0	16388.47	55	2.82	265	16391.34	55.0	16316.51	55	2.25	498
RC2_6_1	15	13275.93	GH	13415.25	15.0	13314.03	15	1.92	1020	13322.77	15.0	**13163.03**	**15**	1.21	1573
RC2_6_2	12	12071.4	GH	11652.71	12.8	12039.89	12	-1.70	1250	11539.21	12.8	**11853.72**	**12**	-2.65	1970
RC2_6_3	11	9978.25	MB	10220.80	11.0	10032.99	11	3.62	1006	10066.43	11.0	**9863.35**	**11**	2.06	1889
RC2_6_4	11	7349.88	MB	7409.35	11.0	7344.31	11	2.46	1069	7274.39	11.0	**7231.64**	**11**	0.59	1993
RC2_6_5	13	11919.72	MB	12224.60	12.8	**12560.43**	**12**	-2.67	1286	12188.40	12.8	12612.91	12	-2.96	1954
RC2_6_6	12	10700.42	LC	12498.61	11.2	12464.98	11	1.76	1242	12405.28	11.2	**12282.52**	**11**	1.00	1963
RC2_6_7	11	11687.04	MB	11510.79	11.0	11347.57	11	4.15	927	11309.89	11.0	**11052.49**	**11**	2.33	1706
RC2_6_8	**11**	**10474.95**	MB	10744.43	11.0	10627.04	11	2.57	894	10617.44	11.0	10488.75	11	1.36	1658
RC2_6_9	11	10113.82	MB	10094.97	11.0	9982.66	11	2.15	895	10060.58	11.0	**9882.71**	**11**	1.80	1661
RC2_610	**11**	**9339.41**	MB	9611.45	11.0	9510.51	11	2.91	900	9500.07	11.0	9340.06	11	1.72	1660
Tot.	2076	798645		823814.02	2076.40	818863.38	2071		37731	811014.16	2076.40	807470.21	2071		65718
Avg.								2.83	629					1.28	1095
< PB						22						30			
#B		29				6						28			

Table 16: Gehring/Homberger VRPTW instances, 600 customers..

	Best known			ALNS 25K						ALNS 50K					
	veh.	cost	References	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)
R1_8_1	**79**	**39612.2**	BVH	37859.00	80.0	37631.40	80	-4.43	684	37756.07	80.0	37492.04	80	-4.69	1010
R1_8_2	**72**	**33548.54**	MB	34705.63	72.0	34435.01	72	3.45	613	34273.25	72.0	33816.69	72	2.16	917
R1_8_3	**72**	**30151.9**	MB	31065.56	72.0	30746.68	72	3.03	645	30593.64	72.0	30317.49	72	1.47	969
R1_8_4	**72**	**26838.04**	MB	29002.34	72.0	28831.80	72	8.06	678	28672.14	72.0	28568.78	72	6.83	1025
R1_8_5	**72**	**34741.53**	MB	36198.65	72.0	36038.57	72	4.19	524	35739.41	72.0	35503.63	72	2.87	809
R1_8_6	**72**	**31737.47**	MB	32820.77	72.0	32757.13	72	3.41	610	32487.44	72.0	32360.07	72	2.36	913
R1_8_7	**72**	**29538.4**	MB	30493.16	72.0	30393.12	72	3.23	644	30089.26	72.0	29979.63	72	1.86	967
R1_8_8	72	28342.64	MB	28803.77	72.0	28622.63	72	1.63	676	28509.27	72.0	**28341.21**	**72**	0.59	1020
R1_8_9	72	34231.38	MB	34961.84	72.0	34856.18	72	2.17	576	34437.83	72.0	**34218.41**	**72**	0.64	864
R1_810	**72**	**31730.45**	MB	33144.45	72.0	32665.95	72	4.46	586	32729.29	72.0	32569.97	72	3.15	879
R2_8_1	**15**	**28440.28**	MB	29209.61	15.0	28923.27	15	2.71	951	29086.28	15.0	28822.48	15	2.27	1811
R2_8_2	15	23335.67	MB	23655.16	15.0	23524.65	15	1.64	1070	23492.15	15.0	**23274.22**	**15**	0.94	1964
R2_8_3	**15**	**17992.25**	MB	18188.06	15.0	18103.52	15	1.09	1127	18137.61	15.0	18078.82	15	0.81	2091
R2_8_4	15	13625.25	MB	13658.48	15.0	13584.57	15	1.82	1213	13525.52	15.0	**13413.79**	**15**	0.83	2322
R2_8_5	**15**	**24611.39**	MB	25479.70	15.0	25260.54	15	3.53	978	25255.01	15.0	25077.09	15	2.62	1803
R2_8_6	**15**	**20697.06**	MB	21104.29	15.0	20969.81	15	1.97	1032	21014.57	15.0	20973.12	15	1.53	1962
R2_8_7	15	17058.3	MB	17114.71	15.0	**16977.49**	**15**	0.81	1119	17128.01	15.0	16980.58	15	0.89	2134
R2_8_8	15	13053.31	MB	13187.89	15.0	13054.95	15	1.87	1254	13063.15	15.0	**12945.52**	**15**	0.91	2365
R2_8_9	**15**	**22588.02**	MB	23303.95	15.0	23138.51	15	3.17	982	23061.61	15.0	22877.21	15	2.10	1849
R2_810	15	21551.26	MB	21372.15	15.0	21240.42	15	1.33	979	21233.28	15.0	**21092.27**	**15**	0.67	1841
C1_8_1	**80**	**25030.36**	M	25184.38	80.0	25184.38	80	0.62	397	25184.38	80.0	25184.38	80	0.62	741
C1_8_2	75	25518.17	GH	25711.25	74.2	25667.72	74	0.68	664	25634.80	74.2	**25536.76**	**74**	0.38	993
C1_8_3	72	25438.6	BVH	25359.87	72.0	24756.97	72	2.96	373	24728.90	72.0	**24629.86**	**72**	0.40	682
C1_8_4	72	24040.47	MB	24256.32	72.0	24118.80	72	1.33	378	24005.77	72.0	**23938.33**	**72**	0.28	706
C1_8_5	80	25166.3	BVH	25166.28	80.0	**25166.28**	**80**	0.00	417	25166.28	80.0	**25166.28**	**80**	0.00	762
C1_8_6	80	25160.9	BVH	25162.17	80.0	**25160.85**	**80**	0.01	560	25162.21	80.0	**25160.85**	**80**	0.01	913
C1_8_7	79	25518.85	GH	25481.02	79.0	**25425.92**	**79**	0.22	623	25449.95	79.0	25428.67	79	0.09	972
C1_8_8	76	25379.85	MB	25740.77	75.2	25622.69	75	1.14	608	25538.76	75.2	**25450.99**	**75**	0.34	930
C1_8_9	73	24713.38	MB	26318.36	72.2	26169.29	72	2.26	575	25673.55	72.2	**25737.46**	**72**	-0.25	868
C1_810	72	29536.81	GH	27097.82	72.0	26382.98	72	5.45	473	26151.75	72.0	**25697.68**	**72**	1.77	770
C2_8_1	**24**	**11654.72**	MB	11678.08	24.0	11665.21	24	0.20	730	11672.47	24.0	11664.00	24	0.15	1238
C2_8_2	**24**	**11422.34**	MB	11456.70	24.0	11428.07	24	0.30	807	11440.98	24.0	11433.46	24	0.16	1397
C2_8_3	**23**	**11554.18**	MB	11312.58	24.0	11184.67	24	-2.09	839	11212.69	24.0	11188.30	24	-2.96	1468
C2_8_4	**23**	**10963.49**	MB	11511.87	23.2	11440.25	23	5.00	955	11180.00	23.2	10999.42	23	1.97	1627
C2_8_5	**24**	**11432.92**	MB	12110.19	24.0	11902.99	24	5.92	896	11565.06	24.0	11451.57	24	1.16	1441
C2_8_6	**24**	**11357.86**	MB	12282.80	24.4	12342.70	24	8.14	812	11909.95	24.2	11403.57	24	4.86	1360
C2_8_7	**24**	**11397.54**	MB	12058.86	24.6	11540.25	24	5.80	881	11871.66	24.4	11412.08	24	4.16	1443
C2_8_8	24	11206.32	MB	12728.62	23.8	13892.26	23	-8.28	860	12371.35	23.8	**13878.40**	**23**	-10.86	1414
C2_8_9	**24**	**11249**	MB	13015.41	24.0	12358.05	24	15.70	897	12446.59	24.0	11650.10	24	10.65	1469
C2_810	**23**	**11284.46**	MB	11837.70	23.8	12103.56	23	4.90	786	11746.59	23.8	12173.74	23	4.10	1358
RC1_8_1	73	31590.23	MB	31990.65	73.0	31851.54	73	2.29	438	31396.64	73.0	**31275.38**	**73**	0.39	720
RC1_8_2	**72**	**39696.2**	GH	29762.99	73.0	29537.14	73	-25.02	608	29377.34	73.0	29172.08	73	-25.99	912
RC1_8_3	**72**	**35577.87**	GH	28634.08	73.0	28466.83	73	-19.52	646	28301.03	73.0	28164.66	73	-20.45	970
RC1_8_4	**72**	**32654.1**	GH	27481.23	73.0	27393.06	73	-15.84	685	27303.22	73.0	27201.39	73	-16.39	1029
RC1_8_5	**73**	**30454.15**	MB	31228.63	73.0	31067.35	73	2.54	578	30742.88	73.0	30548.23	73	0.95	865
RC1_8_6	**73**	**29674.68**	MB	31019.63	73.0	30863.25	73	4.53	573	30749.36	73.0	30511.07	73	3.62	858
RC1_8_7	**72**	**43829.43**	GH	30600.17	73.0	30455.56	73	-30.18	575	30135.52	73.0	30007.82	73	-31.24	868
RC1_8_8	**72**	**43694.6**	GH	30006.93	73.0	29820.15	73	-31.33	580	29603.68	73.0	29547.96	73	-32.25	872
RC1_8_9	**72**	**41816.7**	GH	29918.07	73.0	29812.35	73	-28.45	581	29493.38	73.0	29360.93	73	-29.47	871
RC1_810	**72**	**41182.44**	GH	29518.16	73.0	29373.39	73	-28.32	586	29147.32	73.0	28993.52	73	-29.22	884
RC2_8_1	20	19989.12	MB	20734.14	19.8	21005.11	19	-1.05	1385	20605.53	19.8	**20954.95**	**19**	-1.67	2046
RC2_8_2	17	18099.68	MB	18369.12	17.0	18184.31	17	1.86	1728	18208.97	17.0	**18032.89**	**17**	0.98	2585
RC2_8_3	15	15116.26	MB	15033.15	15.0	**14800.78**	**15**	1.57	1212	14920.27	15.0	14810.81	15	0.81	2172
RC2_8_4	15	11392.25	MB	11592.05	15.0	11402.27	15	1.97	1196	11440.47	15.0	**11368.19**	**15**	0.64	2263
RC2_8_5	**16**	**19105.75**	MB	19293.34	16.4	19214.57	16	0.98	1529	19181.34	16.4	19180.13	16	0.40	2328
RC2_8_6	**15**	**18882.3**	MB	19560.50	15.0	19173.09	15	3.59	1063	19210.08	15.0	19075.89	15	1.74	1918
RC2_8_7	15	17461.44	MB	17798.95	15.0	17519.63	15	2.71	990	17643.28	15.0	**17329.32**	**15**	1.81	1858
RC2_8_8	15	16529.24	MB	16756.53	15.0	16485.06	15	3.26	994	16368.61	15.0	**16226.78**	**15**	0.87	1859
RC2_8_9	15	15823.5	MB	16071.87	15.0	15979.71	15	2.45	989	15902.97	15.0	**15687.20**	**15**	1.38	1825
RC2_810	**15**	**14892.29**	MB	15013.68	15.0	14944.14	15	0.82	1001	15048.01	15.0	14953.29	15	1.05	1873
Tot.	2754	1429914		1381184.08	2762.60	1372619.40	2758		48411	1365178.33	2762.20	1358291.43	2758		81648
Avg.								-0.86	807					-2.07	1361
< PB						15						25			
#B		35				5						22			

Table 17: Gehring/Homberger VRPTW instances, 800 customers..

	Best known			ALNS 25K						ALNS 50K					
	veh.	cost	References	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)	avg. sol.	avg. #veh.	best sol.	best #veh.	avg. gap (%)	avg. time (s)
R110_1	**100**	**54145.31**	MB	55493.78	100.0	55108.89	100	2.49	825	55029.87	100.0	54720.19	100	1.63	1229
R110_2	91	56367.45	GH	54167.93	91.6	57478.64	91	-2.27	698	52844.31	91.6	**55428.79**	**91**	-4.66	1066
R110_3	**91**	**46621.19**	MB	52196.91	91.0	51840.30	91	11.96	435	50296.23	91.0	49634.84	91	7.88	807
R110_4	**91**	**43461.84**	MB	46878.36	91.0	46645.90	91	7.86	435	45626.47	91.0	45303.47	91	4.98	829
R110_5	**91**	**70838.01**	GH	54671.65	92.0	54270.08	92	-22.82	705	53259.01	92.0	53089.15	92	-24.82	1061
R110_6	**91**	**49059.8**	MB	54109.35	91.4	55826.83	91	10.29	541	52485.80	91.4	54555.32	91	6.98	905
R110_7	**91**	**45847.84**	MB	50656.58	91.0	49880.51	91	10.49	429	48869.74	91.0	48141.47	91	6.59	801
R110_8	**91**	**42767.77**	MB	46752.56	91.0	46512.13	91	9.32	452	45286.98	91.0	44853.70	91	5.89	847
R110_9	**91**	**51391.8**	MB	53216.59	92.0	53163.89	92	3.55	706	52139.44	92.0	52015.72	92	1.45	1067
R11010	**91**	**49348.36**	MB	50861.54	92.0	50592.40	92	3.07	674	50007.62	92.0	49769.85	92	1.34	1038
R210_1	**19**	**42922.56**	BSJ	44524.99	19.0	44213.65	19	3.73	1192	43904.40	19.0	43264.68	19	2.29	2083
R210_2	19	34918.49	BSJ	34969.52	19.0	34698.44	19	1.60	1800	34564.64	19.0	**34417.47**	**19**	0.43	2795
R210_3	19	25689.62	BSJ	26067.93	19.0	25964.09	19	2.63	2088	25807.15	19.0	**25400.16**	**19**	1.60	3230
R210_4	19	18858.24	BSJ	18594.33	19.0	18425.77	19	1.43	2289	18477.00	19.0	**18332.77**	**19**	0.79	3480
R210_5	**19**	**37265.32**	BSJ	38149.38	19.0	37773.72	19	2.37	1328	37833.57	19.0	37746.01	19	1.52	2247
R210_6	**19**	**30725.2**	BSJ	31253.33	19.0	30975.00	19	1.72	1453	31007.89	19.0	30778.85	19	0.92	2500
R210_7	19	24363.83	BSJ	24340.48	19.0	24243.39	19	1.45	1923	24228.74	19.0	**23991.71**	**19**	0.99	3009
R210_8	19	18185.38	BSJ	18361.52	19.0	18139.74	19	2.90	2313	18037.86	19.0	**17844.36**	**19**	1.08	3540
R210_9	**19**	**33777.76**	BSJ	35005.84	19.0	34872.05	19	3.64	1345	34496.05	19.0	34349.70	19	2.13	2265
R21010	**19**	**31599.84**	BSJ	32006.08	19.0	31782.57	19	1.29	1645	31803.51	19.0	31682.52	19	0.64	2586
C110_1	**100**	**42478.95**	GH	42478.95	100.0	**42478.95**	**100**	0.00	499	42478.95	100.0	**42478.95**	**100**	0.00	915
C110_2	92	42920.7	BVH	42339.69	91.6	42667.84	91	0.21	798	42222.18	91.6	**42249.60**	**91**	-0.06	1188
C110_3	90	40934.87	MB	41395.50	90.0	40915.89	90	2.52	506	40904.59	90.0	**40376.43**	**90**	1.31	884
C110_4	90	40410.58	MB	40681.78	90.0	40441.12	90	1.76	515	40222.27	90.0	**39980.07**	**90**	0.61	902
C110_5	100	42469.2	BVH	42469.50	100.0	**42469.18**	**100**	0.00	542	42469.18	100.0	**42469.18**	**100**	0.00	968
C110_6	100	42471.3	BVH	42472.69	100.0	**42471.29**	**100**	0.00	678	42471.57	100.0	**42471.29**	**100**	0.00	1103
C110_7	99	42711.39	GH	42726.27	99.0	**42673.51**	**99**	0.12	739	42708.94	99.0	42688.64	99	0.08	1159
C110_8	96	42170.31	MB	42641.48	95.4	42402.12	95	0.67	757	42539.98	95.4	**42359.27**	**95**	0.43	1150
C110_9	91	45386.93	GH	42048.67	91.2	41586.54	91	1.37	645	41774.68	91.2	**41482.00**	**91**	0.71	1005
C11010	**90**	**40894.38**	MB	43409.67	90.0	43132.22	90	6.15	612	42554.17	90.0	42214.60	90	4.06	962
C210_1	**30**	**16879.24**	LL	16905.00	30.0	**16879.24**	**30**	0.15	888	16893.15	30.0	**16879.24**	**30**	0.08	1514
C210_2	**29**	**17228.82**	MB	17446.99	29.4	17677.61	29	1.27	1066	17314.77	29.4	17563.06	29	0.50	1719
C210_3	**29**	**16367.59**	MB	16938.59	30.0	16253.60	30	3.49	971	16446.58	30.0	16109.71	30	0.48	1690
C210_4	29	17153.19	MB	16845.74	29.0	16712.08	29	5.21	1151	16063.32	29.0	**16011.30**	**29**	0.32	1905
C210_5	**30**	**16586.46**	GH	17613.87	30.6	16825.34	30	6.19	964	16888.66	30.4	16596.69	30	1.82	1575
C210_6	30	16371.65	MB	17393.97	30.4	17596.06	30	6.26	1070	16696.06	30.2	**16369.10**	**30**	2.00	1697
C210_7	**31**	**16578.42**	MB	17348.99	31.0	16878.12	31	4.65	978	17057.54	31.0	16590.48	31	2.89	1617
C210_8	**29**	**17219.59**	LC	18921.39	29.6	19122.58	29	9.88	1047	17790.97	29.6	18407.27	29	3.32	1700
C210_9	30	16651.96	MB	17626.12	30.0	16679.15	30	8.17	1104	16999.89	30.0	**16294.72**	**30**	4.33	1771
C21010	**29**	**16178.26**	MB	18856.35	29.0	18447.85	29	16.55	1103	18375.30	29.0	17582.15	29	13.58	1759
RC110_1	**90**	**47143.9**	MB	51246.49	90.0	50976.00	90	8.70	517	49693.36	90.0	48933.68	90	5.41	863
RC110_2	**90**	**44906.58**	MB	47283.88	90.0	46913.77	90	5.29	539	46647.41	90.0	46165.33	90	3.88	904
RC110_3	**90**	**43782.57**	MB	45167.52	90.0	44833.81	90	3.16	562	44408.40	90.0	44014.81	90	1.43	938
RC110_4	**90**	**41917.14**	MB	43355.81	90.0	43144.87	90	3.43	668	42844.52	90.0	42607.34	90	2.21	1071
RC110_5	**90**	**47632.31**	MB	50533.91	90.0	50226.31	90	6.09	431	49082.31	90.0	48934.53	90	3.04	772
RC110_6	**90**	**46391.6**	MB	50436.65	90.0	49703.43	90	8.72	402	49131.04	90.0	48766.98	90	5.91	745
RC110_7	**90**	**46157.71**	MB	49716.92	90.0	49238.95	90	7.71	460	48308.95	90.0	48005.94	90	4.66	806
RC110_8	**90**	**45585.08**	MB	48391.77	90.0	47670.50	90	6.16	396	47416.90	90.0	47122.61	90	4.02	743
RC110_9	**90**	**45405.54**	MB	48343.65	90.0	47930.01	90	6.47	513	46998.60	90.0	46889.79	90	3.51	864
RC11010	**90**	**45041.64**	MB	47210.76	90.0	46716.69	90	4.82	466	46284.90	90.0	46080.51	90	2.76	822
RC210_1	22	30320.41	BSJ	30930.47	21.2	30478.44	21	1.76	1429	30618.08	21.2	**30396.13**	**21**	0.73	2316
RC210_2	19	26592.4	BSJ	26301.14	19.4	**27552.05**	**18**	-4.54	1955	26412.31	19.4	27681.62	18	-4.14	2953
RC210_3	**18**	**20588.38**	BSJ	21313.73	18.0	20983.66	18	3.52	1324	21060.93	18.0	20811.18	18	2.30	2443
RC210_4	18	16480.17	BSJ	16617.79	18.0	16254.55	18	3.81	1345	16499.16	18.0	**16007.59**	**18**	3.07	2544
RC210_5	18	29352.08	LC	29008.22	18.0	28647.57	18	2.26	1249	28610.45	18.0	**28368.48**	**18**	0.85	2198
RC210_6	**18**	**27003.3**	MB	29267.17	18.0	28825.98	18	8.38	1136	29005.97	18.0	28746.61	18	7.42	2035
RC210_7	**18**	**26161.91**	BSJ	27503.47	18.0	27110.84	18	5.13	1106	26958.52	18.0	26765.43	18	3.04	2067
RC210_8	18	24995	BSJ	25445.17	18.0	25211.63	18	1.94	1103	25128.20	18.0	**24961.29**	**18**	0.67	2097
RC210_9	**18**	**23582.89**	MB	24729.65	18.0	24420.99	18	4.86	1129	24417.63	18.0	24113.72	18	3.54	2079
RC21010	**18**	**22481.03**	BSJ	23544.52	18.0	23193.63	18	4.73	1101	23143.27	18.0	23056.75	18	2.95	2066
Tot.	3438	2099741		2157188.54	3443.80	2146751.97	3438		57741	2119549.93	3443.40	2110924.81	3438		95895
Avg.								3.73	962					1.89	1598
< PB						16						22			
#B		38				6						22			

Table 18: Gehring/Homberger VRPTW instances, 1000 customers..

	Optimal		ALNS 25K			
	cost	ref	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
R101***25	**617.1**	KDMSS99	617.1	**617.1**	0.00	3
R102***25	**547.1**	KDMSS99	547.1	**547.1**	0.00	3
R103***25	**454.6**	KDMSS99	454.6	**454.6**	0.00	4
R104***25	**416.9**	KDMSS99	416.9	**416.9**	0.00	4
R105***25	**530.5**	KDMSS99	530.5	**530.5**	0.00	3
R106***25	**465.4**	KDMSS99	465.4	**465.4**	0.00	3
R107***25	**424.3**	KDMSS99	424.3	**424.3**	0.00	4
R108***25	**397.3**	KDMSS99	397.3	**397.3**	0.00	4
R109***25	**441.3**	KDMSS99	441.3	**441.3**	0.00	3
R110***25	**444.1**	KDMSS99	444.1	**444.1**	0.00	4
R111***25	**428.8**	KDMSS99	428.8	**428.8**	0.00	4
R112***25	**393**	KDMSS99	393.0	**393.0**	0.00	4
C101***25	**191.3**	KDMSS99	191.3	**191.3**	0.00	4
C102***25	**190.3**	KDMSS99	190.3	**190.3**	0.00	4
C103***25	**190.3**	KDMSS99	190.3	**190.3**	0.00	4
C104***25	**186.9**	KDMSS99	186.9	**186.9**	0.00	4
C105***25	**191.3**	KDMSS99	191.3	**191.3**	0.00	4
C106***25	**191.3**	KDMSS99	191.3	**191.3**	0.00	4
C107***25	**191.3**	KDMSS99	191.3	**191.3**	0.00	4
C108***25	**191.3**	KDMSS99	191.3	**191.3**	0.00	5
C109***25	**191.3**	KDMSS99	191.3	**191.3**	0.00	4
RC101***25	**461.1**	KDMSS99	461.1	**461.1**	0.00	4
RC102***25	**351.8**	KDMSS99	351.8	**351.8**	0.00	4
RC103***25	**332.8**	KDMSS99	332.8	**332.8**	0.00	4
RC104***25	**306.6**	KDMSS99	306.6	**306.6**	0.00	4
RC105***25	**411.3**	KDMSS99	411.3	**411.3**	0.00	4
RC106***25	**345.5**	KDMSS99	345.5	**345.5**	0.00	4
RC107***25	**298.3**	KDMSS99	298.3	**298.3**	0.00	4
RC108***25	**294.5**	KDMSS99	294.5	**294.5**	0.00	4
R201***25	**463.3**	L99	463.3	**463.3**	0.00	4
R202***25	**410.5**	L99	410.5	**410.5**	0.00	4
R203***25	**391.4**	L99	391.4	**391.4**	0.00	4
R204***25	**355**	C03	355.2	**355.0**	0.06	6
R205***25	**393**	L99	393.0	**393.0**	0.00	5
R206***25	**374.4**	CR99	374.4	**374.4**	0.00	5
R207***25	**361.6**	KLM01	361.6	**361.6**	0.00	5
R208***25	**328.2**	FDGG04	328.2	**328.2**	0.00	11
R209***25	**370.7**	KLM01	371.5	**370.7**	0.21	6
R210***25	**404.6**	CR99	404.6	**404.6**	0.00	5
R211***25	**350.9**	KLM01	350.9	**350.9**	0.00	6
C201***25	**214.7**	L99	214.7	**214.7**	0.00	7
C202***25	**214.7**	L99	214.7	**214.7**	0.00	8
C203***25	**214.7**	L99	214.7	**214.7**	0.00	8
C204***25	**213.1**	CR99	214.4	**213.1**	0.59	8
C205***25	**214.7**	L99	214.7	**214.7**	0.00	8
C206***25	**214.7**	L99	214.7	**214.7**	0.00	7
C207***25	**214.5**	L99	214.5	**214.5**	0.00	9
C208***25	**214.5**	L99	214.5	**214.5**	0.00	7
RC201***25	**360.2**	L99	360.2	**360.2**	0.00	4
RC202***25	**338**	CR99	338.0	**338.0**	0.00	4
RC203***25	**326.9**	FDGG04	326.9	**326.9**	0.00	4
RC204***25	**299.7**	FDGG04	299.7	**299.7**	0.00	5
RC205***25	**338**	L99	338.0	**338.0**	0.00	4
RC206***25	**324**	KLM01	324.0	**324.0**	0.00	4
RC207***25	**298.3**	KLM01	298.3	**298.3**	0.00	5
RC208***25	**269.1**	C03	269.1	**269.1**	0.00	6
Tot.	18551.0		18553.2	18551.0		276
Avg.					0.02	5
< PB				0		
#B	56			56		

Table 19: Solomon VRPTW instances with 25 customers, comparison to exact solutions (distances and travel times are truncated to one decimal and traveled distance is minimized). The table should be read as the preceding tables. The abbreviations in the *ref* column refers to the following papers: *C03* – Chabrier [10], *CR99* – Cook and Rich [14], *DP03* – Danna and Le Pape [20], *FDGG04* – Feillet [24], *IV03* – Irnich and Villeneuve [34], *KLM01* – Kallehauge et al. [35], *KDMSS99* – Kohl et al. [36] and *L99* – Larsen [39].

	Optimal		ALNS 25K			
	cost	ref	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
R101***50	**1044**	KDMSS99	1044.0	**1044.0**	0.00	9
R102***50	**909**	KDMSS99	909.0	**909.0**	0.00	10
R103***50	**772.9**	KDMSS99	772.9	**772.9**	0.00	10
R104***50	**625.4**	KDMSS99	626.1	**625.4**	0.12	11
R105***50	**899.3**	KDMSS99	899.8	**899.3**	0.05	9
R106***50	**793**	KDMSS99	793.0	**793.0**	0.00	10
R107***50	**711.1**	KDMSS99	711.1	**711.1**	0.00	10
R108***50	**617.7**	CR99	617.7	**617.7**	0.00	11
R109***50	**786.8**	KDMSS99	786.8	**786.8**	0.00	10
R110***50	**697**	KDMSS99	697.0	**697.0**	0.00	10
R111***50	**707.2**	L99	707.2	**707.2**	0.00	10
R112***50	**630.2**	L99	635.1	635.0	0.77	11
C101***50	**362.4**	KDMSS99	362.4	**362.4**	0.00	9
C102***50	**361.4**	KDMSS99	361.4	**361.4**	0.00	11
C103***50	**361.4**	KDMSS99	361.4	**361.4**	0.00	11
C104***50	**358**	KDMSS99	358.0	**358.0**	0.00	12
C105***50	**362.4**	KDMSS99	362.4	**362.4**	0.00	10
C106***50	**362.4**	KDMSS99	362.4	**362.4**	0.00	10
C107***50	**362.4**	KDMSS99	362.4	**362.4**	0.00	10
C108***50	**362.4**	KDMSS99	362.4	**362.4**	0.00	11
C109***50	**362.4**	KDMSS99	362.4	**362.4**	0.00	12
RC101***50	**944**	KDMSS99	944.0	**944.0**	0.00	9
RC102***50	**822.5**	KDMSS99	822.8	**822.5**	0.04	10
RC103***50	**710.9**	KDMSS99	710.9	**710.9**	0.00	10
RC104***50	**545.8**	KDMSS99	545.8	**545.8**	0.00	10
RC105***50	**855.3**	KDMSS99	855.3	**855.3**	0.00	10
RC106***50	**723.2**	KDMSS99	723.2	**723.2**	0.00	9
RC107***50	**642.7**	KDMSS99	643.7	**642.7**	0.16	10
RC108***50	**598.1**	KDMSS99	598.1	**598.1**	0.00	10
R201***50	**791.9**	L99	795.8	**791.9**	0.49	13
R202***50	**698.5**	L99	698.5	**698.5**	0.00	14
R203***50	**605.3**	C03	608.2	605.9	0.49	15
R204***50	**506.4**	IV03	506.4	**506.4**	0.00	24
R205***50	**690.1**	C03	698.2	696.7	1.17	15
R206***50	**632.4**	C03	634.0	**632.4**	0.25	16
R207***50	-	-	576.1	**576.1**	0.01	22
R208***50	-	-	489.6	**487.7**	0.39	29
R209***50	**600.6**	C03	602.5	**600.6**	0.32	15
R210***50	**645.6**	C03	648.3	**645.6**	0.42	16
R211***50	**535.5**	IV03	549.8	543.3	2.67	25
C201***50	**360.2**	L99	360.2	**360.2**	0.00	25
C202***50	**360.2**	CR99	360.2	**360.2**	0.00	27
C203***50	**359.8**	CR99	359.8	**359.8**	0.00	27
C204***50	**350.1**	KLM01	350.1	**350.1**	0.00	29
C205***50	**359.8**	CR99	359.8	**359.8**	0.00	30
C206***50	**359.8**	CR99	359.8	**359.8**	0.00	26
C207***50	**359.6**	CR99	359.6	**359.6**	0.00	27
C208***50	**350.5**	CR99	350.5	**350.5**	0.00	28
RC201***50	**684.4**	L99	684.8	684.8	0.06	12
RC202***50	**613.6**	FDGG04	613.6	**613.6**	0.00	12
RC203***50	**555.3**	C03	555.3	**555.3**	0.00	15
RC204***50	**444.2**	DP03	444.2	**444.2**	0.00	19
RC205***50	**630.2**	FDGG04	630.2	**630.2**	0.00	12
RC206***50	**610**	FDGG04	610.3	**610.0**	0.05	13
RC207***50	**558.6**	FDGG04	558.6	**558.6**	0.00	16
RC208***50	-	-	497.9	**481.8**	3.33	24
Tot.			32560.9	32519.7		841
Avg.					0.19	15

Table 20: Solomon VRPTW instances with 50 customers, comparison to exact solutions .

	Optimal		ALNS 25K			
	cost	ref	avg. sol.	best sol.	avg. gap (%)	avg. time (s)
R101	**1637.7**	KDMSS99	1638.6	**1637.7**	0.05	30
R102	**1466.6**	KDMSS99	1467.7	1467.6	0.08	33
R103	**1208.7**	CR99	1208.9	**1208.7**	0.01	34
R104	**971.5**	IV03	977.1	976.0	0.58	34
R105	**1355.3**	KDMSS99	1355.8	**1355.3**	0.03	31
R106	**1234.6**	L99	1234.6	**1234.6**	0.00	33
R107	**1064.6**	L99	1068.2	**1064.6**	0.34	33
R108	-	-	943.5	**933.7**	1.05	36
R109	**1146.9**	CR99	1150.2	**1146.9**	0.29	31
R110	**1068**	CR99	1083.1	1075.6	1.41	33
R111	**1048.7**	CR99	1049.2	**1048.7**	0.05	33
R112	-	-	952.2	**948.6**	0.38	35
C101	**827.3**	KDMSS99	827.3	**827.3**	0.00	29
C102	**827.3**	KDMSS99	827.3	**827.3**	0.00	32
C103	**826.3**	KDMSS99	826.3	**826.3**	0.00	34
C104	**822.9**	KDMSS99	822.9	**822.9**	0.00	36
C105	**827.3**	KDMSS99	827.3	**827.3**	0.00	30
C106	**827.3**	KDMSS99	827.3	**827.3**	0.00	31
C107	**827.3**	KDMSS99	827.3	**827.3**	0.00	31
C108	**827.3**	KDMSS99	827.3	**827.3**	0.00	32
C109	**827.3**	KDMSS99	827.3	**827.3**	0.00	34
RC101	**1619.8**	KDMSS99	1629.8	**1619.8**	0.61	28
RC102	**1457.4**	CR99	1475.1	1463.5	1.22	30
RC103	**1258**	CR99	1272.2	1267.0	1.13	31
RC104	-	-	1132.8	**1132.6**	0.01	33
RC105	**1513.7**	KDMSS99	1514.2	1513.8	0.04	30
RC106	-	-	1376.1	**1373.9**	0.16	29
RC107	**1207.8**	IV03	1213.0	1209.3	0.43	30
RC108	**1114.2**	IV03	1124.6	**1114.2**	0.94	31
R201	**1143.2**	KLM01	1153.9	1148.5	0.94	45
R202	-	-	1041.0	**1036.9**	0.40	54
R203	-	-	876.5	**872.4**	0.47	60
R204	-	-	731.5	**731.3**	0.03	67
R205	-	-	952.4	**949.8**	0.27	58
R206	-	-	880.6	**880.6**	0.00	61
R207	-	-	796.4	**794.0**	0.30	72
R208	-	-	703.1	**701.2**	0.27	86
R209	-	-	860.2	**855.8**	0.52	60
R210	-	-	914.0	**908.4**	0.61	59
R211	-	-	758.3	**752.3**	0.80	67
C201	**589.1**	CR99	589.1	**589.1**	0.00	69
C202	**589.1**	CR99	589.1	**589.1**	0.00	74
C203	**588.7**	KLM01	588.7	**588.7**	0.00	80
C204	**588.1**	IV03	588.1	**588.1**	0.00	84
C205	**586.4**	CR99	586.4	**586.4**	0.00	76
C206	**586**	CR99	586.0	**586.0**	0.00	72
C207	**585.8**	CR99	585.8	**585.8**	0.00	74
C208	**585.8**	KLM01	585.8	**585.8**	0.00	74
RC201	**1261.8**	KLM01	1272.3	1262.6	0.84	42
RC202	**1092.3**	C03	1097.4	1095.8	0.47	46
RC203	-	-	937.6	**923.7**	1.50	56
RC204	-	-	788.1	**785.8**	0.29	68
RC205	**1154**	C03	1154.0	**1154.0**	0.00	45
RC206	-	-	1062.5	**1051.1**	1.08	52
RC207	-	-	976.2	**966.6**	0.99	55
RC208	-	-	790.5	**777.3**	1.70	65
Tot.			54752.7	54579.5		2649
Avg.					0.36	47

Table 21: Solomon VRPTW instances with 100 customers, comparison to exact solutions .

	Best known			ALNS 25K					ALNS 50K				
	n	type	cost	avg. sol.	best sol.	avg. gap (%)	best above B.K	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	best above B.K	avg. time (s)
P01	50	C	**524.61**	524.61	**524.61**	0.00	0.00	12	524.61	**524.61**	0.00	0.00	21
P02	75	C	**835.26**	841.81	838.87	0.78	0.43	20	839.62	**835.26**	0.52	0.00	38
P03	100	C	**826.14**	828.18	**826.14**	0.25	0.00	46	826.99	**826.14**	0.10	0.00	85
P04	150	C	**1028.42**	1037.43	1031.23	0.88	0.27	96	1034.20	1029.56	0.56	0.11	176
P05	199	C	**1291.29**	1309.36	1298.92	1.40	0.59	124	1306.63	1297.12	1.19	0.45	233
P06	50	CD	**555.43**	555.43	**555.43**	0.00	0.00	12	555.43	**555.43**	0.00	0.00	21
P07	75	CD	**909.68**	913.03	**909.68**	0.37	0.00	19	911.78	**909.68**	0.23	0.00	36
P08	100	CD	**865.94**	867.65	**865.94**	0.20	0.00	42	866.97	**865.94**	0.12	0.00	78
P09	150	CD	**1162.55**	1169.06	1164.24	0.56	0.15	86	1167.68	1163.68	0.44	0.10	160
P10	199	CD	**1395.85**	1408.19	1404.17	0.88	0.60	116	1410.27	1405.88	1.03	0.72	219
P11	120	C	**1042.11**	1042.37	1042.12	0.03	0.00	73	1042.46	1042.12	0.03	0.00	132
P12	100	C	**819.56**	819.56	**819.56**	0.00	0.00	43	819.56	**819.56**	0.00	0.00	79
P13	120	CD	**1541.14**	1543.77	1542.86	0.17	0.11	61	1543.54	1542.86	0.16	0.11	113
P14	150	CD	**866.37**	866.37	**866.37**	0.00	0.00	40	866.37	**866.37**	0.00	0.00	73
Tot.			13664	13726.83	13690.13			789	13716.09	13684.21			1464
Avg.						0.39	0.15	56			0.31	0.11	105
< PB				0					0				
#B	14			7					8				

Table 22: Christofides et al. CVRP problems [13]. The column *type* indicates if the problem is capacity constrained (*C*) or both capacity and duration constrained (*CD*). The column *best above B.K* indicates how much the best solution found differs from the best known solution from the literature (in percent). The best known solutions where obtained from Cordeau et al. [16].

	Best known				ALNS 25K					ALNS 50K				
	n	type	cost	ref	avg. sol.	best sol.	avg. gap (%)	best above B.K	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	best above B.K	avg. time (s)
KELLY01	240	C	**5627.54**	MB	5667.04	5660.88	0.70	0.59	193	5662.57	5650.91	0.62	0.42	393
KELLY02	320	C	**8447.92**	MB	8499.27	8478.73	0.61	0.36	321	8487.94	8469.32	0.47	0.25	672
KELLY03	400	C	**11036.22**	MB	11067.48	11045.81	0.28	0.09	482	11052.72	11047.01	0.15	0.10	1015
KELLY04	480	C	**13624.52**	MB	13752.13	13635.31	0.94	0.08	654	13748.50	13635.31	0.91	0.08	1328
KELLY05	200	C	**6460.98**	TK	6479.80	6478.09	0.29	0.26	306	6482.49	6466.68	0.33	0.09	629
KELLY06	280	C	**8412.8**	MB	8507.29	8415.67	1.12	0.03	418	8543.30	8416.13	1.55	0.04	876
KELLY07	360	C	10195.56	MB	10273.80	10231.34	0.90	0.35	464	10265.15	**10181.75**	0.82	-0.14	941
KELLY08	440	C	**11663.55**	MB	11804.08	11721.35	1.20	0.50	499	11766.07	11713.62	0.88	0.43	1011
KELLY09	255	CD	**583.39**	MB	591.75	584.48	1.43	0.19	225	590.33	585.14	1.19	0.30	437
KELLY10	323	CD	**742.03**	MB	753.48	749.47	1.54	1.00	305	751.36	748.89	1.26	0.92	616
KELLY11	399	CD	**918.45**	MB	933.42	926.63	1.63	0.89	371	926.57	922.70	0.88	0.46	761
KELLY12	483	CD	**1107.19**	MB	1129.53	1125.11	2.02	1.62	458	1125.22	1119.06	1.63	1.07	911
KELLY13	252	CD	**859.11**	MB	878.22	876.01	2.22	1.97	142	874.24	864.68	1.76	0.65	285
KELLY14	320	CD	**1081.31**	MB	1107.97	1096.92	2.47	1.44	194	1103.53	1095.40	2.06	1.30	393
KELLY15	396	CD	**1345.23**	MB	1370.94	1355.91	1.91	0.79	236	1366.23	1359.94	1.56	1.09	468
KELLY16	480	CD	**1622.69**	MB	1652.00	1639.81	1.81	1.05	279	1645.67	1639.11	1.42	1.01	549
KELLY17	240	CD	**707.79**	MB	713.76	710.36	0.84	0.36	153	710.59	708.90	0.39	0.16	304
KELLY18	300	CD	**998.73**	MB	1008.11	1003.20	0.94	0.45	196	1007.84	1002.42	0.91	0.37	387
KELLY19	360	CD	**1366.86**	MB	1380.49	1377.52	1.00	0.78	227	1377.88	1374.24	0.81	0.54	449
KELLY20	420	CD	**1821.15**	MB	1843.08	1828.35	1.20	0.40	245	1834.70	1830.80	0.74	0.53	488
Tot.			88623		89413.66	88940.91			6369	89322.91	88832.02			12914
Avg.							1.25	0.66	318			1.02	0.48	646
< PB					0					1				
#B	19				0					1				

Table 23: Golden et al. CVRP problems ([30]). The best known solutions where obtained from Cordeau et al. [16], *MB* refers to the heuristic by Mester and Bräysy [42] (the results are not given in [42], but can be found in [16]), *TK* refers to the heuristic by Tarantilis and Kiranoudis [57].

	Best known			ALNS 25K					ALNS 50K				
	n	cost	ref	avg. sol.	best sol.	avg. gap (%)	best above B.K	avg. time (s)	avg. sol.	best sol.	avg. gap (%)	best above B.K	avg. time (s)
CVRP_L_21	560	**16212.83**	EST	16488.67	16296.21	1.70	0.51	869	16391.23	16224.81	1.10	0.07	1735
CVRP_L_22	600	14641.64	ORTR	14737.97	14638.37	0.73	-0.02	569	14644.06	**14631.08**	0.09	-0.07	1168
CVRP_L_23	640	**18801.13**	EST	19155.50	18925.36	1.88	0.66	1097	19112.56	18837.49	1.66	0.19	2268
CVRP_L_24	720	**21389.43**	EST	22024.22	21652.78	2.97	1.23	1259	21913.83	21522.48	2.45	0.62	2739
CVRP_L_25	760	17053.26	EST	17170.49	17082.81	1.59	0.17	650	17115.78	**16902.16**	1.26	-0.89	1320
CVRP_L_26	800	**23977.74**	EST	24577.43	24084.92	2.50	0.45	1425	24405.05	24014.09	1.78	0.15	3081
CVRP_L_27	840	17651.6	ORTR	17833.67	17749.35	1.25	0.55	723	17769.75	**17613.22**	0.89	-0.22	1504
CVRP_L_28	880	**26566.04**	EST	27315.94	26651.15	2.82	0.32	1692	27172.63	26791.72	2.28	0.85	3441
CVRP_L_29	960	**29154.34**	EST	30117.04	29487.26	3.30	1.14	1887	29976.86	29405.60	2.82	0.86	3921
CVRP_L_30	1040	**31742.64**	EST	32828.86	32133.28	3.42	1.23	2192	32607.06	31968.33	2.72	0.71	4348
CVRP_L_31	1120	**34330.94**	EST	35617.70	34962.16	3.75	1.84	2395	35472.51	34770.34	3.33	1.28	5003
CVRP_L_32	1200	**36919.24**	EST	37989.05	37401.49	2.90	1.31	2750	37818.65	37377.35	2.44	1.24	5321
Tot.		288441		295856.55	291065.13			17509	294399.98	290058.65			35849
Avg.						2.40	0.78	1459			1.90	0.40	2987
< PB					1					3			
#B		9			0					3			

Table 24: Li et al. CVRP problems [40]. *EST* refers to a solution found by hand by Li et al [40] (the instances are highly symmetrical which makes it easy to construct good solutions by hand). *ORTR* refers to a solution found by a heuristic by Li et al. [40].

Part III

Exact methods

Chapter 7

Introduction to exact methods

7.1 Introduction

Solving NP-hard optimization problems to optimality is a topic, that has challenged researchers almost since the beginning of computer history (long before the concept of NP-hardness was discovered). Significant progress has been made in the recent decades, but for many problem types only fairly small instances can be solved. Vehicle routing problems belong to a class of problems that has proved to be difficult to solve. Only moderately sized problems can be solved to optimality consistently.

The topic of this chapter and the following is quite different from that studied in part II and so is the goal of the methods developed. In part II we studied heuristics, and an objective that always was kept in mind in this part of the thesis was the applicability of the heuristics to real life problems.

In this part of the thesis, we are not concerned about the methods we develop should be applicable to real life problems. The purposes of the work presented in this part is mainly to

- Enhance our knowledge about the PDPTW. This include investigating which formulations of the problem that appear to be best suited for solving the problem to optimality and finding new valid inequalities for the problem.
- Provide the research community with knowledge about optimal solutions for the PDPTW. This can be used to evaluate the performance of heuristics for the problem.

In this chapter we review some of the methods used for solving NP-hard optimization problems to optimality. We review the methods that has been used in Chapter 8 and 9, namely *branch-and-cut* (BAC) and *branch-and-price* (BAP) and their combination *branch-cut-and-price* (BCP). It is assumed that the reader is familiar with the branch-and-bound paradigm, if not, for example Wolsey [1998] gives an introduction to the subject.

7.2 Linear programming based lower bounds

The structure of this section is to a certain degree inspired from Ralphs and Galati [2005] and the notation has been taken from same paper.

In what follows we assume that a integer linear program (ILP) is to be solved and we assume that the problem is a minimization problem. We can write the problem as:

$$z_{IP} = \min_{x \in \mathbb{Z}^n} \left\{ c^T x \,|\, Ax \geq b \right\}$$

where $c \in \mathbb{Q}^n$, $A \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$. Such a problem is often solved using a branch-and-bound method.

When solving NP-hard optimization problems exactly, one often resort to branch-and-bound methods. In order to construct a branch-and-bound algorithm for solving z_{IP}, one needs a lower bound to z_{IP} - preferably one that can computed efficiently. One way to obtain a lower bound is to solve the linear relaxation of z_{IP}

$$z_{LP} = \min_{x \in \mathbb{R}^n} \left\{ c^T x \,|\, Ax \geq b \right\} \tag{7.1}$$

as z_{LP} is minimizing over a superset $\mathbb{Z}^n$, it is clearly a valid lower bound to z_{IP}. Let $\mathcal{Q} = \{x \in \mathbb{R}^n \,|\, Ax \geq b\}$ be the polyhedron of feasible solutions to z_{LP}. We can express the set of feasible solutions to z_{IP} as $\mathcal{F} = \mathcal{Q} \cap \mathbb{Z}^n$. Now let $conv\,(\cdot)$ denote the convex hull. From Wolsey [1998], proposition 1.1 and 1.2 we know that $conv\,(\mathcal{F})$ is a polyhedron and that the extreme points of $conv(X)$ always lie in X, this gives us the the second equality sign in

$$z_{IP} = \min_{x \in \mathbb{Z}^n} \left\{ c^T x \,|\, Ax \geq b \right\} = \min_{x \in \mathcal{F}} \left\{ c^T x \right\} = \min_{x \in conv(\mathcal{F})} \left\{ c^T x \right\}$$

The fact that $\mathcal{P} = conv(\mathcal{F})$ is a polyhedron means that Z_{IP} in theory can be solved as linear program — in practice, this does not give us a way to solve Z_{IP} though, as we for interesting problems do not know the inequalities defining $\mathcal{P}$ and even if we knew the set of inequalities it could be of exponential size. We can try to approximate $\mathcal{P}$ though. This is what is being done in a branch-and-cut algorithm in order to get better lower bounds, we will return to this a little later.

To illustrate the concepts, consider the following ILP

$$\min -2x_1 + 3x_2$$

subject to

$$2x_1 + x_2 \geq 8 \tag{7.2}$$
$$-3x_1 + x_2 \geq -12 \tag{7.3}$$
$$x_1 - 4x_2 \geq -20 \tag{7.4}$$
$$-x_1 + 2x_2 \geq -1 \tag{7.5}$$
$$x_1, x_2 \in \mathbb{Z} \tag{7.6}$$

The polyhedron $\mathcal{Q}$ is in this case defined by equation (7.2)–(7.5) and

$$x_1, x_2 \in \mathbb{R} \tag{7.7}$$

The polyhedron $\mathcal{Q}$ is illustrated in Figure 7.1 (upper left). We find that $z_{LP} = -3.8$ and the optimal point in $\mathcal{Q}$ is $(x_1^*, x_2^*) = (4.6, 1.8)$ (see Figure 7.1, lower right). The set $\mathcal{F} = \mathcal{Q} \cap \mathbb{Z}^2$ and the polyhedron $conv(\mathcal{F})$ are easily obtained in this simple case and is illustrated in Figure 7.1 (upper right and lower left). We find that $z_{IP} = -2$ and the optimal point in $\mathcal{F}$ is $(x_1^*, x_2^*) = (4, 2)$. We note that z_{LP} indeed is a lower bound to z_{IP}.

It is possible to use the lower bound z_{LP} in a branch and bound algorithm, but often a tighter bound is desirable. As we saw above it is possible to close the gap between z_{LP} and z_{IP} completely by adding all the inequalities from $conv(\mathcal{F})$ to the linear programming problem. This leads to the term *valid inequality.* The definition of a valid inequality (from definition 8.1 in Wolsey [1998]) is

Definition: An inequality $\pi x \geq \pi_0$ is a valid inequality for $\mathcal{F} \subseteq \mathbb{R}^n$ if $\pi x \geq \pi_0$ for all $x \in \mathcal{F}$.

In other words, a valid inequality should not cut away any feasible integer points. We are of course interested in inequalities that do cut away fractional solutions. Given an optimal solution x^* to the LP relaxation that is fractional we want to find a valid inequality that cuts away x^*. The problem of finding such an inequality is called a *separation problem* as we want to find an inequality that separates x^* from $\mathcal{F}$. An algorithm that finds such an inequality is called a *separation algorithm.*

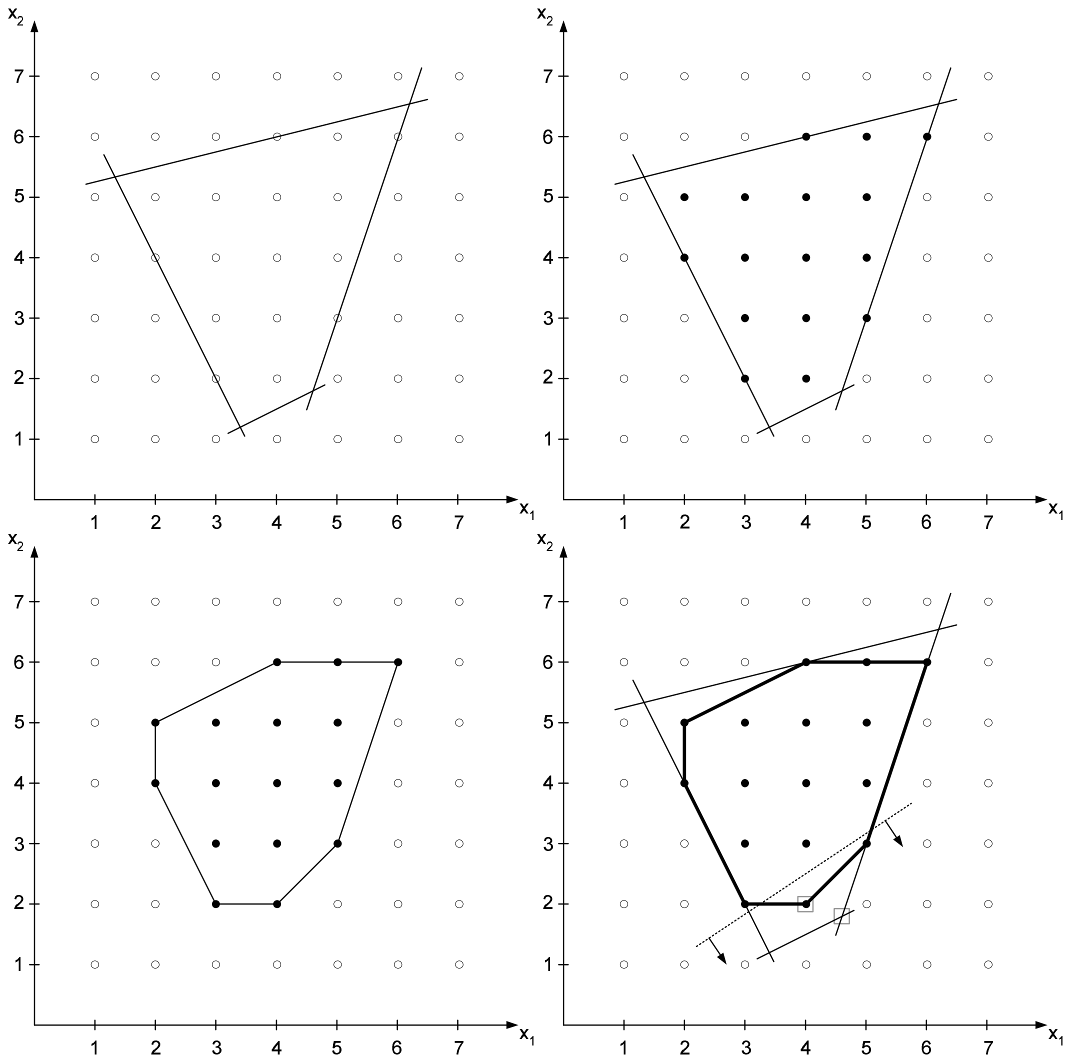


Figure 7.1: Polyhedrons. The upper left figure illustrates the polyhedron $\mathcal{Q}$ defined by equations (7.2)–(7.5) and (7.7). The upper right figure illustrates $\mathcal{F} = \mathcal{Q} \cap \mathbb{Z}^2$, the elements of $\mathcal{F}$ are the black dots. The figure at the bottom left illustrates the convex hull of $\mathcal{F}$. The figure in the bottom right illustrates both $\mathcal{Q}$, $\mathcal{F}$ and $conv(\mathcal{F})$, the figure also show the objective (the dotted line) and the optimal LP and IP solutions (grey rectangles).

In Figure 7.2 (left) we have added the valid inequality

$$-2x_1 + x_2 \geq -7 \tag{7.8}$$

Adding this inequalities separates the old optimal LP solution $x^* = (4.6, 1.8)$ from $\mathcal{F}$. The new optimal LP solution is $x^* = (4\frac{1}{3}, 1\frac{2}{3})$ with objective $-3\frac{2}{3}$, which is a slightly better lower bound. Adding the inequalities

$$-x_1 + x_2 \geq -2 \tag{7.9}$$

$$x_2 \geq 2 \tag{7.10}$$

once again separates x^* from $\mathcal{F}$ and the LP relaxation now gives the optimal solution to the integer problem, $x^* = (4, 2)$, $z_{IP} = z_{LP} = -2$, this is illustrated in Figure 7.2 (right). This example shows that it is not necessary to have a complete description of the convex hull of $\mathcal{F}$ to get the IP optimal solution using the LP relaxation. A description of the convex hull around the IP optimal point is enough. In this case the inequalities (7.9) and (7.10) are enough. We also see that the inequality (7.8) is redundant because of inequalities (7.9) and (7.3). Inequalities (7.9) and (7.10) are examples of the so called *facet defining inequalities* of the polyhedron conv($\mathcal{F}$) while (7.8) is defining a *face* of the polyhedron conv($\mathcal{F}$). The facet defining inequalities are the strongest valid inequalities. The proper definition of facets and faces are given in Definition 9.5 in Wolsey [1998]

Definition

1. F defines a *face* of the polyhedron P if $F = \{x \in P : \pi x = \pi_0\}$ for some valid inequality $\pi x \geq \pi_0$ of P.
2. F is a *facet* of P if F is a face of P and $\dim(F) = \dim(P) - 1$.
3. If F is a face of P with $F = \{x \in P : \pi x = \pi_0\}$, the valid inequality $\pi x \geq \pi_0$ is said to *represent* or *define* the face.

To give a more intuitive sense of faces and facets we can say, that for a 2-dimensional polyhedron like the one in our figure, the faces are lines and extreme points of the polyhedron while the facets are the lines defining the polyhedron. For a 3-dimensional polyhedron the faces are the planes plus the extreme lines and extreme points of the polyhedron, facets are the planes of the polyhedron.

7.2.1 Cutting plane algorithm

Valid inequalities give us a an algorithmic framework for solving IP problems. Our starting point is the linear relaxation (7.1). By using the definition of $\mathcal{Q}$ from section 7.2 we can write it as

$$z_{LP} = \min \left\{ c^T x \,|x \in \mathcal{Q} \right\} \tag{7.11}$$

the cutting plane algorithm works by iteratively adding valid inequalities to $\mathcal{Q}$ to make it approximate conv(F) better and better. The algorithm follows the following steps (assuming that the IP has a feasible solution)

1. set $t = 0$, $\mathcal{Q}^t = \mathcal{Q}$
2. solve the linear program
$$z^t = \min \left\{ c^T x \,\middle|x \in \mathcal{Q}^t \right\}$$
3. let x^* be the corresponding optimal solution.
4. if x^* is integer then
   we have found the optimal solution to the IP problem - STOP.
5. Find an inequality $\pi^t x \geq \pi_0^t$ that separates x^* from $\mathcal{F}$.

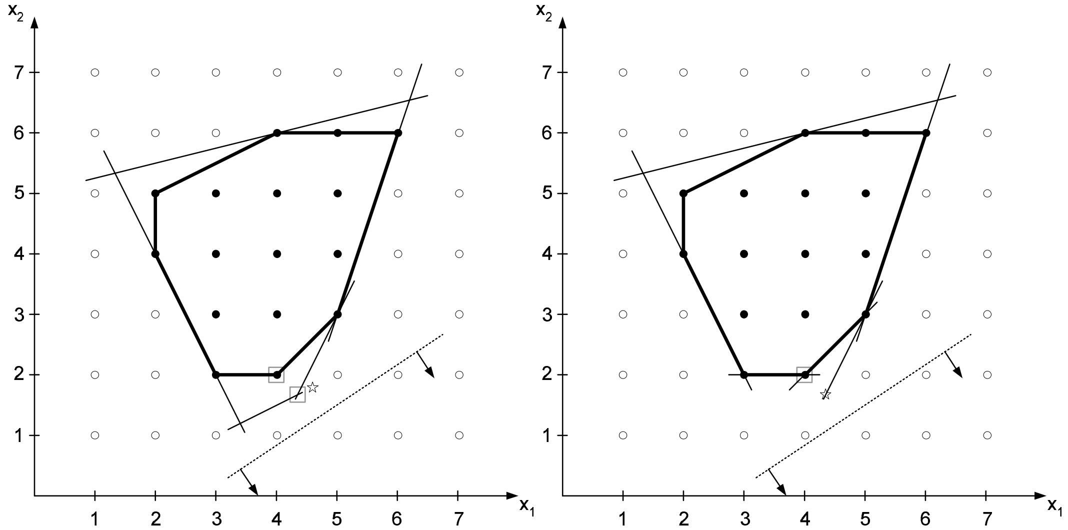


Figure 7.2: Valid inequalities. In the left figure the inequality (7.8) has been added to (7.2)–(7.5), yielding a better lower bound. The LP optimal solution is marked with a square while the old LP optimal solution is marked with a star. In the right figure the inequalities (7.9) and (7.10) have been added and the LP relaxation now gives the IP optimal solution, the star marks the old LP optimal solution.

6. Set $\mathcal{Q}^{t+1} = \mathcal{Q}^t \cap \{x \in \mathbb{R}^n \,|\pi^t x \geq \pi_0^t\}$, set $t = t + 1$
7. Goto step 2

Step 5 is of course the most complex part of the algorithm, but *Gomory's fractional cutting plane algorithm* (GFCPA) provides a way of generating a violated valid inequality and it can be proved that the GFCPA finds the IP optimal solution in a finite number of steps (see [Nemhauser and Wolsey, 1988, Theorem 3.8]). The convergence of the GFCPA is very slow though and the algorithm is not really useful in practice.

Instead one can separate inequalities from a family of inequalities, known to be useful for the particular IP problem (e.g. one could use the *comb inequalities* if solving the TSP). When no more violated inequalities can be found the algorithm must stop and the problem is instead solved by branch and bound using the tighter lower bound provided by $z_{LP} = \min\left\{c^T x \,|x \in \mathcal{Q}^t\right\}$ instead of $z_{LP} = \min\left\{c^T x \,|x \in \mathcal{Q}\right\}$.

7.2.2 Branch-and-cut

Branch-and-cut extends on the cutting plane algorithm idea from the preceding section. The idea in branch-and-cut is simply to generate valid, violated inequalities throughout the branch and bound tree and not only in the root node. The valid inequalities are typically chosen from some preselected families of valid inequalities and in each node there is a trade off between improving the lower bound as much as possible versus processing the node as fast as possible. Thus one will often stop generating valid inequalities in a node if the improvements of the lower bound has been small for a number of iterations. In that case it may be better to branch and then try to generate more valid inequalities in the child nodes. It is important to note that some cuts are globally valid - they can be used throughout the branch and bound tree, even if detected in a child node deep in the tree, while other cuts are locally valid - they can only be used in the node where they were discovered and in its child nodes. In the branch-and-cut algorithm presented in this paper only globally valid cuts are used.

The branch and cut paradigm has been successful for many problem types, most notable is probably the development in TSP branch-and-cut methods Applegate et al. [2003].

7.3 Introduction to branch-and-price

The preceding section introduced one technique for getting strong lower bounds and shortly discussed how to use these lower bounds in a branch and bound framework. This section introduces the Dantzig-Wolfe decomposition technique for integer programs and investigates how it can be used within a branch-and-bound method. Dantzig Wolfe decomposition was originally introduced for linear programs by Dantzig and Wolfe [1960].

The branch-and-price paradigm relies on two concepts. The first concept is decomposition that transforms the original or *compact formulation* into a model that contains many columns, but typically fewer rows than the original formulation. The new formulation is often denoted an *extensive formulation.*

The second concept is *column generation.* In order to solve the linear relaxation of the extensive formulation one does not generate the entire model as it typically is very large - the number of variables often grow exponentially in the size of the original problem. Instead columns are generated dynamically using a technique known as column generation. When the lower bound within a branch-and-bound framework is solved using dynamic column generation the resulting branch and bound algorithm is called *branch-and-price* or *IP column generation.* Adding a branch and bound search on top of a linear programming relaxation based on column generation might seem straightforward, but the approach has some pitfalls. An example of such a pittfall is how to create the subproblems when branching such that they doesn't change the structure of the pricing problem.

This section only gives a short introduction to column generation and branch-and-price, more information can be found in Wolsey [1998], Desrosiers and Lübbecke [2005], Ralphs and Galati [2005], Sigurd [2004], Vanderbeck [2000]. The description given in this chapter follows that of Wolsey [1998] and Sigurd [2004].

7.3.1 Decomposition

Consider an integer programming problem of the form (*compact formulation*)

$$\min c^T x \tag{7.12}$$

subject to

$$\begin{aligned} Ax \geq \; & b & (7.13)\\ Dx \geq \; & d & (7.14)\\ x \in \; & \mathbb{N}_0^n & (7.15) \end{aligned}$$

Where A is an $m_A \times n$ matrix and D is an $m_D \times n$ matrix and all elements are assumed to be rationals, $b \in \mathbb{Q}^{m_A}$ and $d \in \mathbb{Q}^{m_D}$ are vectors. Assume that the polyhedron $\left\{x \in \mathbb{R}_+^n : Dx \geq d\right\}$ is bounded (see Sigurd [2004] for the unbounded case). Then the set $X = \{x \in \mathbb{N}_0^n : Dx \geq d\}$ contains a finite number of elements $\{p_\omega\}_{\omega \in \Omega}$ and we can write the set X as

$$X = \left\{ x \in \mathbb{R}_+^n : x = \sum_{\omega \in \Omega} \lambda_\omega p_\omega, \sum_{\omega \in \Omega} \lambda_\omega = 1, \lambda_\omega \in \{0,1\}, \forall \omega \in \Omega \right\}$$

Substituting for x in the compact formulation (7.12)–(7.15) leads to the *extensive formulation*

$$\min c^T \left(\sum_{w \in \Omega} \lambda_\omega p_\omega \right) \tag{7.16}$$

subject to

$$A\left(\sum_{w\in\Omega}\lambda_\omega p_\omega\right) \geq \quad b \tag{7.17}$$

$$\sum_{\omega\in\Omega}\lambda_\omega = \quad 1 \tag{7.18}$$

$$\lambda_\omega \in \{0,1\} \qquad \forall\omega\in\Omega \tag{7.19}$$

Defining $c_\omega = c^T p_\omega$ and $a_\omega = Ap_\omega$ for $\omega \in \Omega$ we get

$$\min \sum_{w\in\Omega} c_\omega \lambda_\omega \tag{7.20}$$

subject to

$$\sum_{\omega\in\Omega} a_\omega\lambda_\omega \geq \quad b \tag{7.21}$$

$$\sum_{\omega\in\Omega}\lambda_\omega = \quad 1 \tag{7.22}$$

$$\lambda_\omega \in \{0,1\} \qquad \forall\omega\in\Omega \tag{7.23}$$

The extensive formulation contains fewer rows (inequalities) than the compact formulation but it typically contains many more columns (variables). The real benefit of the extensive formulation is that its LP relaxation often is better (tighter) than the LP relaxation of the compact formulation. To see this consider the two polyhedra $\mathcal{Q}_c = \left\{x \in \mathbb{R}^n_+ : Ax \geq b, Dx \geq d\right\}$ and $\mathcal{Q}_e = \left\{x \in \mathbb{R}^n_+ : Ax \geq d, x \in \text{conv}(X)\right\}$. $\mathcal{Q}_c$ is the polyhedron that the linear relaxation of the compact formulation optimizes over, while $\mathcal{Q}_e$ is the polyhedron that the linear relaxation of the extensive formulation optimizes over. It is clear that $\mathcal{Q}_e \subseteq \mathcal{Q}_c$ as

$$\text{conv}(X) = \text{conv}\left(\left\{x \in \mathbb{N}_0^+ : Dx \geq d\right\}\right) \subseteq \left\{x \in \mathbb{R}^n_+ : Dx \geq d\right\}$$

If $\text{conv}(X) = \left\{x \in \mathbb{R}^n_+ : Dx \geq d\right\}$ then the extensive and the compact formulation give the same lower bound. This is the case when all the extreme points of $\left\{x \in \mathbb{R}^n_+ : Dx \geq d\right\}$ are integer and the polyhedron is said to have the *integrality property*.

The price one has to pay for getting a tighter lower bound is that the set X must be known. In Section 7.3.2 we show that it is not necessary to have an explicit definition of X in the the model.

The decomposition described above is in particular useful if the matrix D has a block diagonal structure, that is

$$D = \begin{pmatrix} D^1 & & \\ & \ddots & \\ & & D^\Delta \end{pmatrix}$$

The matrices $D^\delta, \delta = 1, \ldots, \Delta$ are $m_\delta \times n_\delta$ matrices and the d=$(d^1, \ldots, d^\Delta)$ where $d^\delta \in \mathbb{Q}^{m_\delta}$. In that case we can consider the smaller, independent polyhedrons $X^\delta = \left\{x \in \mathbb{N}_0^{n_\delta} : D^\delta x \geq d^\delta\right\}$ that are bounded if X is and therefore contains a finite number of elements $\left\{p_\omega^\delta\right\}_{\omega\in\Omega^\delta}$. We can write X^δ as

$$X^\delta = \left\{x \in \mathbb{R}_+^{n_\delta} : x = \sum_{\omega\in\Omega^\delta}\lambda_\omega^\delta p_\omega^\delta, \sum_{\omega\in\Omega^\delta}\lambda_\omega^\delta = 1, \lambda_\omega^\delta \in \{0,1\}, \forall\omega\in\Omega^\delta\right\}$$

Let ${p'}_\omega^\delta = \left(0, \ldots, 0, \left(p_\omega^\delta\right)^T, 0, \ldots, 0\right)^T$ where there are $\sum_{i=1}^{\delta-1} n^i$ leading zeros and $\sum_{i=\delta+1}^{\Delta} n^i$ trailing zeros . The polyhedron X can be written as $X = X^1 \times \ldots \times X^\Delta$ and we can express any point

x in X as

$$x = \sum_{\delta=1}^{\Delta} \left(\sum_{\omega \in \Omega^\delta} \lambda_\omega^\delta {p'}_\omega^\delta \right)$$

substituting into (7.12)–(7.15) yields

$$\min c^T \left(\sum_{\delta=1}^{\Delta} \left(\sum_{\omega \in \Omega^\delta} \lambda_\omega^\delta {p'}_\omega^\delta \right) \right) \tag{7.24}$$

subject to

$$A \left(\sum_{\delta=1}^{\Delta} \left(\sum_{\omega \in \Omega^\delta} \lambda_\omega^\delta {p'}_\omega^\delta \right) \right) \geq \quad b \tag{7.25}$$

$$\sum_{\omega \in \Omega^\delta} \lambda_\omega^\delta = \quad 1 \qquad \forall \delta \in \{1, \ldots \Delta\} \tag{7.26}$$

$$\lambda_\omega^\delta \in \{0, 1\} \qquad \forall \delta \in \{1, \ldots \Delta\}, \forall \omega \in \Omega^\delta \tag{7.27}$$

Defining $c_\omega^\delta = c^T {p'}_\omega^\delta$ and $a_\omega^\delta = A{p'}_\omega^\delta$ this simplifies to

$$\min \sum_{\delta=1}^{\Delta} \sum_{\omega \in \Omega^\delta} c_\omega^\delta \lambda_\omega^\delta \tag{7.28}$$

subject to

$$\sum_{\delta=1}^{\Delta} \sum_{\omega \in \Omega^\delta} a_\omega^\delta \lambda_\omega^\delta \geq \quad b \tag{7.29}$$

$$\sum_{\omega \in \Omega^\delta} \lambda_\omega^\delta = \quad 1 \qquad \forall \delta \in \{1, \ldots \Delta\} \tag{7.30}$$

$$\lambda_\omega^\delta \in \{0, 1\} \qquad \forall \delta \in \{1, \ldots \Delta\}, \forall \omega \in \Omega^\delta \tag{7.31}$$

The model (7.28)–(7.31) has fewer variables than model (7.20)–(7.23) that did not take advantage of the block diagonal structure (see for example Sigurd [2004]).

If all the blocks in the diagonal block matrix are identical, then by selecting Ω^1 as representative of the points in the X^δ sets, the model can be simplified to

$$\min \sum_{\omega \in \Omega^1} c_\omega^1 \lambda_\omega^1 \tag{7.32}$$

subject to

$$\sum_{\omega \in \Omega^1} a_\omega^1 \lambda_\omega^1 \geq \ b \tag{7.33}$$

$$\sum_{\omega \in \Omega^1} \lambda_\omega^1 = \Delta \tag{7.34}$$

$$\lambda_\omega^1 \in \mathbb{N}_0 \qquad \forall \omega \in \Omega^1 \tag{7.35}$$

7.3.2 Column generation

Column generation is a technique for solving large scale linear programming problems. When using column generation we are not using all variables explicitly in the model we are solving, but only a subset. Variables are generated dynamically when necessary by using the properties of the simplex algorithm. Column generation can be used for any linear programming problem, but it is particularly useful for linear programs with a huge number of variables as the ones arising from the extensive models obtained in section 7.3.1. In this section we are going to see how column generation works for the linear relaxation of (7.20)–(7.23), we refer to Sigurd [2004] for a more throughout exposition.

The linear relaxation of (7.20)–(7.23) is:

$$\min \sum_{w \in \Omega} c_\omega \lambda_\omega \tag{7.36}$$

subject to

$$\sum_{\omega \in \Omega} a_\omega \lambda_\omega \geq b \tag{7.37}$$

$$\sum_{\omega \in \Omega} \lambda_\omega = 1 \tag{7.38}$$

$$0 \leq \lambda_\omega \leq 1 \qquad \forall \omega \in \Omega \tag{7.39}$$

Thus the linear relaxion considers a linear combination of the elements from X. Consider looking at a reduced set of columns $\bar{\Omega} \subseteq \Omega$ such that $|\bar{\Omega}|$ is much smaller that $|\Omega|$. $\bar{\Omega}$ must be chosen such that the linear program

$$\min \sum_{w \in \bar{\Omega}} c_\omega \lambda_\omega \tag{7.40}$$

subject to

$$\sum_{\omega \in \bar{\Omega}} a_\omega \lambda_\omega \geq b \tag{7.41}$$

$$\sum_{\omega \in \bar{\Omega}} \lambda_\omega = 1 \tag{7.42}$$

$$0 \leq \lambda_\omega \leq 1 \qquad \forall \omega \in \bar{\Omega} \tag{7.43}$$

has a feasible solution. If it is difficult to select a subset of columns such that the linear program has a feasible solution, then one can generate one or more *dummy columns* that has very high cost, but constitute a feasible solution. In order to proceed one first need to consider how the simplex algorithm solves a linear program like

$$\min c^T x \tag{7.44}$$

subject to

$$Ax \geq b \tag{7.45}$$

$$x \in \mathbb{R}^n_+ \tag{7.46}$$

The simplex algorithm maintains a *basic feasible solution* that as the name implies is a feasible solution to the LP, but not necessarily optimal. In each iteration of the simplex algorithm a new column is chosen to *enter the basis*. If the new column should have a chance of improving the basic feasible solution, it must have negative *reduced cost* c_i^π

$$c_i^\pi = c_i - \pi A_i$$

where π is the current dual variables associated with the constraints (7.45) and A_i is the ith column in A. The typical approach is to select the column with minimum reduced cost, that is, the column

$$\arg \min_{i \in \{1,\ldots,n\}} \{c_i - \pi A_i\}$$

when no column with negative reduced cost is found, then the simplex algorithm has reached the optimal solution to (7.44)–(7.46).

Returning to column generation for the relaxed, decomposed problem (7.36)–(7.39), it is now clear, that after solving the reduced problem (7.40)–(7.43) we can use the dual vector π to see if more columns should be added to $\bar{\Omega}$. We simply have to calculate the reduced costs of all the columns in $\Omega \setminus \bar{\Omega}$. If one of these columns has negative reduced cost then it is added to $\bar{\Omega}$ and (7.40)–(7.43) is resolved. Frequently one will add the column with the most negative reduced cost. If no column in $\Omega \setminus \bar{\Omega}$ has negative reduced cost then the solution to (7.40)–(7.43) is optimal for (7.36)–(7.39) as well.

The above description require us to know column A_ω for all $\omega \in \Omega$ or at least to have a function that can generate all these columns on demand. This is impractical for most problems. Instead we need an *oracle* that given the dual vector π can return a column with negative reduced cost or tell if such a column does not exist. The oracle solves the following problem

$$\min c^T x - \pi A x \tag{7.47}$$

subject to

$$x \in X \tag{7.48}$$

or alternatively

$$\min c^T x - \pi A x \tag{7.49}$$

subject to

$$Dx \geq d \tag{7.50}$$

$$x \in \mathbb{N}_0^n \tag{7.51}$$

The problem (7.49)–(7.51) is called the *pricing problem* while the problem (7.40)–(7.43) is called the *master problem*. The pricing problem is often a hard problem in itself.

One decomposition of the VRPTW decomposes the problem into a master problem that is a set-partitioning problem and a pricing problem that is a elementary shortest path problem with time windows and capacity constraints which is a NP-hard problem. In Chapter 9 several decompositions of the PDPTW is considered.

7.3.3 Branch-and-price

Section 7.3.2 showed how an LP lower bound for the extensive formulation of an integer program could be obtained. Sometimes the LP solution happens to be integer and the original IP is solved. But in general the LP solution will be fractional. The paradigm *branch-and-price* or *IP column generation* deals with how to obtain an integer solution when the LP relaxation is fractional.

The obvious approach is to use branch and bound, where branching is performed on the λ_ω variables, that is, if λ_ω is fractional for some $\omega \in \Omega$ then two branches are created, one where $\lambda_w = 1$ and one where $\lambda_\omega = 0$. One problem with this approach is that it creates highly unbalanced branch-and-bound trees as the $\lambda_\omega = 0$ branch most often does not change the problem much - we are excluding one column out of millions of columns. The second problem with the branching rule is that it can create difficulties for the pricing problem. The first case is generally easy to handle while the second case is problematic - it changes the structure of the pricing problem - when imposing $\lambda_\omega = 0$ the pricing problem is no longer allowed to generate column ω. Depending on the pricing problem this can make it much harder to solve.

To avoid these problems one prefers a branching scheme that is *compatible* with the pricing problem. This can often be obtained by branching on the variables of the compact formulation. For the VRPTW Dumas et al. [1991] for example proposed to branch on the arcs in the compact formulation, this information was easily transfered to the subproblem.

The topic of branching in IP column generation is discussed further in the literature mentioned in section 7.3 and in Chapter 9 it is discussed how branching can be done for an IP column generation algorithm for the PDPTW.

7.3.4 Branch-and-cut-and-price

It is possible to combine the branch-and-cut and branch-and-price paradigms to obtain even stronger lower bounds. For problems of the vehicle routing family this has been proposed by Kohl [1995], Desaulniers et al. [1998], Kohl et al. [1999]. The approach described by Kohl [1995], Kohl et al. [1999] allows us to generate cuts based on the x-variables in the compact formulation (7.12)–(7.15). Such a cut can be expressed as $\alpha x \geq \beta$ where $\beta \in \mathbb{Q}$ and $\alpha^T \in \mathbb{Q}^n$. The x vector corresponding to the current solution of the master problem is simple to obtain as $x = \sum_{\omega \in \bar{\Omega}} \lambda_\omega p_\omega$. When a cut in the original variables has been identified, it can be added to the master problem by substituting for x. The cut in the λ variables is: $\alpha \sum_{\omega \in \Omega} \lambda_\omega p_\omega \geq \beta$ or alternatively $\sum_{w \in \Omega} \alpha_\omega \lambda_\omega \geq \beta$ where $\alpha_\omega = \alpha p_\omega$. Adding this row to the master problem changes the objective of the pricing problem to

$$\min c^T x - \pi A x - \nu \alpha x \tag{7.52}$$

while the equations (7.50)–(7.51) remains the same. ν is the dual variable corresponding to the new row. What happens with the pricing problem is that some of coefficients of the variables in the objective function are changed. This usually means that the pricing problem occurring when adding cuts can be solved by the same pricing algorithm as was used to solve the pricing problem when no cuts were added. Such a cut is called *robust* in the terminology introduced by Poggi de Aragão and Uchoa [2003]. As we are going to see in Chapter 9 it is not always the case that the change in objective is harmless to the pricing problem.

Poggi de Aragão and Uchoa [2003] present a different way of handling cuts expressed in the variables of the original formulation. In their decomposition they keep the original variables x and can introduce cuts in a direct way. The drawback of this approach is that larger linear programs must be solved compared to the approach outlined above.

Jepsen et al. [2005] experiments with some classes of cuts derived from the *clique inequality* for the set-partitioning problem, that operate directly on the λ variables. This changes the structure of the pricing problem and makes them harder to solve, but has a significant impact on the lower bounds. Computational tests on the VRPTW are promising.

7.3.5 Further topics

This section gives pointers to the literature for further topics within column generation.

- **Alternative decomposition**. An alternative way of decomposing the compact formulation is proposed by Desrosiers et al. [1995]. This decomposition is done as in linear programming, using Minkowski's Theorem by decomposing by conv(X). Vanderbeck [2000] compares this approach to the decomposition presented in section 7.3.1. The two decompositions give the same lower bounds.

- **Relation to Lagrangian relaxation**. It has long been known that performing Lagrangian relaxation on the compact formulation (7.12)–(7.15) by relaxing the constraints $Ax \geq b$ gives the same lower bound as performing Dantzig-Wolfe decomposition where the constraints $Ax \geq b$ is kept in the master problem. This was shown by Geoffrion [1974]. Research have been carried in the recent years to combine the two approaches. This topic is considered by Huisman et al. [2005] while Kallehauge et al. [2006] compares a Lagrangian relaxation approach to a column generation approach.

- **Stabilization**. It has been observed that the convergence of column generation algorithms can be very slow. The observation made is that dual variable initially fluctuates violently and their initial values seem almost random. This implies that worthless columns are generated early on in the process. The fluctuation in dual variables only slowly dies out, and it is typically only towards the end of the column generation process that useful columns (the ones ending up in the optimal LP solution) are generated. To alleviate this problem, *stabilized column generation* has been proposed. Stabilized column generation works by limiting how much the dual variables can change. This can be done by selecting a current "guess" of the

dual variables. Setting the dual variable to a value far from the guess is penalized, typically by a piecewise linear function. This causes the dual variables to stay close to the guess. At times the guesses are updated, moving them toward the current value of the dual in question, and the penalty functions can be modified as well. A different approach is suggested by Rousseau et al. [2003]. Given an optimal primal solution they consider the polydron $\mathcal{D}$ containing all optimal dual solutions. They show how to find a set of different extreme points of $\mathcal{D}$ (each point corresponds to a feasible, optimal dual solution) and produces an interior point in $\mathcal{D}$ by a creating a convex combination of the set of extreme points. This interior point is more stable than the extreme point of $\mathcal{D}$ returned by the LP when using an unstabilized approach and the computational experiments suggest that method is comparable to the approach based on penalties functions while requiring fewer parameters.
Some references to literature about stabilization in column generation are du Merle et al. [1999], Sigurd and Ryan [2003], Rousseau et al. [2003], Amor et al. [2004] and Oukil et al. [2004]. Significant speed ups are reported when using stabilization.

- **Implementation tricks**. Many implementation tricks for speeding up column generation algorithms have been proposed. Some of these are: solving the pricing problem heuristically, adding more than one column in each iteration, only keeping a limited set of the columns generated in the linear programming model. These and many more tricks are described in Desaulniers et al. [2001] and Lübbecke and Desrosiers [2005].

Chapter 8

Models and a Branch-and-Cut Algorithm for Pickup and Delivery Problems with Time Windows

Models and Branch-and-Cut Algorithms for Pickup and Delivery Problems with Time Windows

STEFAN ROPKE*†
JEAN-FRANÇOIS CORDEAU†
GILBERT LAPORTE†

June 29, 2006

Abstract

In the pickup and delivery problem with time windows (PDPTW), capacitated vehicles must be routed to satisfy a set of transportation requests between given origins and destinations. In addition to capacity and time window constraints, vehicle routes must also satisfy pairing and precedence constraints on pickups and deliveries. This paper introduces two new formulations for the PDPTW and the closely related dial-a-ride problem (DARP) in which a limit is imposed on the elapsed time between the pickup and the delivery of a request. Several families of valid inequalities are introduced to strengthen these two formulations. These inequalities are used within branch-and-cut algorithms which have been tested on several sets of instances for both the PDPTW and the DARP. Instances with up to eight vehicles and 96 requests (192 nodes) have been solved to optimality.

Keywords: pickup and delivery; time windows; valid inequalities; branch-and-cut.

*DIKU, University of Copenhagen, Denmark
†Canada Research Chair in Distribution Management, HEC Montréal, Canada

1 Introduction

In the *Pickup and Delivery Problem* (PDP), capacitated vehicles must be routed to satisfy a set of transportation requests between given origins and destinations. Each route must start and finish at a common depot and satisfy pairing and precedence constraints: for each request, the origin must precede the destination, and both locations must be visited by the same vehicle. The PDP arises naturally in several contexts such as urban courier services and door-to-door transportation systems for the elderly and the disabled. In most applications, time windows restrict the time at which each pickup and delivery location may be visited by a vehicle. This gives rise to the *PDP with Time Windows* (PDPTW). In the case of passenger transportation, additional constraints may also be present to reduce customer dissatisfaction. In particular, ride time constraints are often imposed to limit the time spent by a passenger in the vehicle. The resulting problem is called the *Dial-a-Ride Problem* (DARP).

Both the PDP and PDPTW are generalizations of the classical *Vehicle Routing Problem* (VRP) and are thus $\mathcal{NP}$-hard. As a result, the development of solution methods for these problems has focused on heuristics (see, e.g., Desaulniers et al., 2002; Cordeau et al., 2006). Nevertheless, when the problem is sufficiently constrained, it is possible to obtain optimal solutions within reasonable computation time. For instance, dynamic programming has been used successfully to solve the single-vehicle PDP with or without time windows (Psaraftis, 1980, 1983; Desrosiers et al., 1986). For the multiple-vehicle case, column generation approaches have been proposed. The first such method was introduced by Dumas et al. (1991) who addressed the PDPTW. Their set-partitioning formulation is solved by a branch-and-price method in which columns of negative reduced-cost are generated by a dynamic programming algorithm similar to that of Desrosiers et al. (1986) for the single-vehicle case. The method has been successful in solving instances with tight capacity constraints and a small number of requests per route. Several arc elimination rules have also been proposed to reduce the size of the problem. A similar approach was later developed by Savelsbergh and Sol (1998) who used a column management mechanism to reduce the size of the master problem, and construction and improvement heuristics to accelerate the solution of the pricing subproblem.

Another solution methodology that has proven successful for solving the PDP is branch-and-cut. The single-vehicle case without time windows was first studied by Ruland and Rodin (1997) who introduced several families of valid inequalities that are also valid for the PDPTW and will thus be described in more detail in Section 3. Branch-and-cut has also been used to solve the more general *Precedence-Constrained Asymmetric Traveling Salesman Problem* (PCATSP) in which each node may have multiple predecessors. Valid inequalities and a branch-and-cut algorithm for this problem have been developed, respectively, by Balas et al. (1995) and Ascheuer et al. (2000b). A branch-and-cut algorithm for the capacitated multiple-vehicle PDP and PDPTW was later described by Lu and Dessouky (2004). Their formulation contains a polynomial number of constraints and uses two-index flow variables, but relies on extra variables to impose pairing and precedence constraints. Instances with up to five vehicles and 25 requests were solved optimally with this approach. More recently, Cordeau (2006) has developed a branch-and-cut algorithm for the DARP. It is based on a three-index formulation with a polynomial number of constraints. It uses several families of valid inequalities that are either adaptations of existing inequalities for the TSP and the VRP, or new inequalities which take advantage of the structure of the problem. Most of these inequalities are valid for the PDPTW and will also be described in Section 3. This approach was capable of solving instances with up to four vehicles and 32 requests.

In this paper, we introduce new branch-and-cut algorithms for the classical version of the PDPTW, as defined in Desaulniers et al. (2002), and the closely related DARP. We make three contributions. First, we propose two new formulations for the PDPTW which, unlike the formulation of Cordeau (2006), have an exponential number of constraints, but lead to more efficient solution algorithms because they contain fewer variables and provide tighter bounds. Second, we introduce new valid inequalities combining the pickup and delivery structure of the problem with either the vehicle capacity constraints or the time window constraints. Third, we report computational experiments on several sets of test instances and show that our approach is capable of solving some instances with up to eight vehicles and 96 requests.

The remainder of the paper is organized as follows. Section 2 formally defines the PDPTW and introduces two formulations of the problem. Section 3 describes the valid inequalities used in the branch-and-cut algorithms which are then introduced in Section 4. Computational results are reported in Section 5, followed by conclusions in the last section.

2 Formulations of the PDPTW

Let n denote the number of requests to satisfy. The PDPTW can be defined on a directed graph $G = (N, A)$ with node set $N = \{0, \ldots, 2n+1\}$ and arc set A. Nodes 0 and $2n+1$ represent the origin and destination depots (which may have the same location) while subsets $P = \{1, \ldots, n\}$ and $D = \{n+1, \ldots, 2n\}$ represent pickup and delivery nodes, respectively. With each request i are thus associated a pickup node i and a delivery node $n+i$. With each node $i \in N$ are associated a load q_i and a non-negative service duration d_i satisfying $d_0 = d_{2n+1} = 0$, $q_0 = q_{2n+1} = 0$, and for $i = 1, \ldots, n$, $q_i \geq 0$ and $q_{n+i} = -q_i$. An unlimited fleet of identical vehicles with capacity Q is available to serve the requests. With each arc $(i, j) \in A$ are associated a routing cost c_{ij} and a travel time t_{ij}. A time window $[e_i, l_i]$ is also associated with every node $i \in P \cup D$, where e_i and l_i represent the earliest and latest time, respectively, at which service may start at node i. The depot nodes may also have time windows $[e_0, l_0]$ and $[e_{2n+1}, l_{2n+1}]$ representing the earliest and latest times, respectively, at which the vehicles may leave from and return to the depot. We assume that the triangle inequality holds both for routing costs and travel times. For any node subset $S \subseteq N$, define its complement $\bar{S} = N \setminus S$. Finally, to impose pairing and precedence constraints, it is convenient to define the set $\mathcal{S}$ of all node subsets $S \subseteq N$ such that $0 \in S$, $2n+1 \notin S$ and there is at least one request i for which $i \notin S$ and $n+i \in S$.

For each arc $(i, j) \in A$ let x_{ij} be a binary variable equal to 1 if and only if a vehicle travels directly from node i to node j. For each node $i \in P \cup D$ let B_i be the time at which service begins at node i, and Q_i be the vehicle load upon leaving node i.

The PDPTW can be formulated as the following mixed-integer program:

$$\text{(PDPTW1)} \qquad \text{Minimize} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \tag{1}$$

subject to

$$\begin{aligned}
\sum_{i \in N} x_{ij} &= 1 & \forall j \in P \cup D \quad (2)\\
\sum_{j \in N} x_{ij} &= 1 & \forall i \in P \cup D \quad (3)\\
\sum_{i,j \in S} x_{ij} &\leq |S| - 2 & \forall S \in \mathcal{S} \quad (4)\\
B_j &\geq (B_i + d_i + t_{ij}) x_{ij} & \forall i \in N, j \in N \quad (5)\\
Q_j &\geq (Q_i + q_j) x_{ij} & \forall i \in N, j \in N \quad (6)\\
e_i \leq B_i &\leq l_i & \forall i \in N \quad (7)\\
\max\{0, q_i\} \leq Q_i &\leq \min\{Q, Q + q_i\} & \forall i \in N \quad (8)\\
x_{ij} &\in \{0, 1\} & \forall i \in N, j \in N. \quad (9)
\end{aligned}$$

The objective function (1) minimizes the total routing cost. Constraints (2) and (3) require each node to be visited exactly once. Consistency of the time and load variables is ensured through constraints (5) and (6). The respect of time windows and vehicle capacity is then ensured through constraints (7) and (8). Under the assumption that $d_i + t_{i,n+i} > 0$ for every request i, constraints (5) and (7) also ensure that no subtours exist in the solution.

Finally, inequalities (4) are *precedence constraints* (see Ruland and Rodin, 1997) which guarantee that for each user i, node $n+i$ is visited after node i and both nodes are visited by the same vehicle. These constraints were originally proposed in a single-vehicle context but they apply directly to the multi-vehicle case because route feasibility conditions are the same in both cases. In multi-vehicle problems constraints (4) do not only enforce precedence relations but, because of the definition of S, they also ensure that the two nodes of the same request are on the same route. Indeed, suppose that a feasible integer solution contains a path $(0, k_1, \ldots, k_r, n+i)$ where $k_j \neq i$, $\forall j$, i.e., a path connects the origin depot to node $n+i$ without visiting node i. In this case, the set $S = \{0, k_1, \ldots, k_r, n+i\}$ clearly belongs to $\mathcal{S}$ and leads to a violation of the associated inequality (4). It is also worth pointing out that because $x(S) = |S| - 1 - x(\delta^-(S))$, inequality (4) can be written equivalently as $x(\delta^-(S)) \geq 1$.

By introducing variables L_i representing the ride time of each user i, and denoting by L the maximum ride time, the DARP can be modeled by introducing the following constraints:

$$\begin{aligned}
L_i &= B_{n+i} - (B_i + d_i) & \forall i \in P \quad (10)\\
t_{i,n+i} \leq L_i &\leq L & \forall i \in P. \quad (11)
\end{aligned}$$

Formulation (1)-(9) is non-linear because of constraints (5) and (6). Introducing constants M_{ij} and W_{ij}, these constraints can, however, be linearized as follows:

$$B_j \geq B_i + d_i + t_{ij} - M_{ij}(1 - x_{ij}) \qquad \forall i \in N, j \in N \tag{12}$$

$$Q_j \geq Q_i + q_j - W_{ij}(1 - x_{ij}) \qquad \forall i \in N, j \in N. \tag{13}$$

The validity of these constraints is ensured by setting $M_{ij} \geq \max\{0, l_i + d_i + t_{ij} - e_j\}$ and $W_{ij} \geq \min\{Q, Q + q_i\}$. As shown by Desrochers and Laporte (1991), constraints (12) and (13), for a given pair $i, j \in N$, can be lifted as follows by taking the reverse arc (j, i) into account:

$$B_j \geq B_i + d_i + t_{ij} - M_{ij}(1 - x_{ij}) + (M_{ij} - d_i - t_{ij} - \max\{d_j + t_{ji}, e_i - l_j\})x_{ji} \tag{14}$$

$$Q_j \geq Q_i + q_j - W_{ij}(1 - x_{ij}) + (W_{ij} - q_i - q_j)x_{ji}. \tag{15}$$

In the case of the DARP, lifting (14) is, however, invalid because of constraints (10) and (11) which put additional restrictions on the time variables B_i.

As suggested by Desrochers and Laporte, bounds on the time variables can also be strengthened as follows:

$$B_i \geq e_i + \sum_{j \in N \setminus \{i\}} \max\{0, e_j - e_i + d_j + t_{ij}\}x_{ji} \tag{16}$$

$$B_i \leq l_i - \sum_{j \in N \setminus \{i\}} \max\{0, l_i - l_j + d_i + t_{ij}\}x_{ij}. \tag{17}$$

Similarly, bounds on load variables Q_i can be strengthened as follows:

$$Q_i \geq \max\{0, q_i\} + \sum_{j \in N \setminus \{i\}} \max\{0, q_j\}x_{ji} \tag{18}$$

$$Q_i \leq \min\{Q, Q + q_i\} - (Q - \max_{j \in N \setminus \{i\}}\{q_j\} - q_i)x_{0i} - \sum_{j \in N \setminus \{i\}} \max\{0, q_j\}x_{ij}. \tag{19}$$

A formulation with fewer variables can be obtained by replacing constraints (5)-(8) with *rounded capacity inequalities* (see, e.g., Naddef and Rinaldi, 2002) and *infeasible path elimination constraints* (see, e.g., Ascheuer et al., 2000a). For any subset $S \subseteq P \cup D$, define $q(S) = \sum_{i \in S} q_i$. A lower bound on the number of times vehicles must enter and leave S in order to visit all nodes in the set is then provided by $\lceil |q(S)|/Q \rceil$. Denote by $\mathcal{R}$ the set of infeasible paths with respect to time windows, and for each path $R \in \mathcal{R}$, let $A(R) \subset A$ be the set of arcs in this path. With these definitions, the PDPTW can be reformulated as follows:

$$\text{(PDPTW2)} \qquad \text{Minimize} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \tag{20}$$

subject to

$$\sum_{i \in N} x_{ij} = 1 \qquad \forall j \in P \cup D \tag{21}$$

$$\sum_{j \in N} x_{ij} = 1 \qquad \forall i \in P \cup D \tag{22}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 2 \qquad \forall S \in \mathcal{S} \tag{23}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - max\left\{1, \frac{\lceil |q(S)| \rceil}{Q}\right\} \qquad \forall S \subseteq N \setminus \{0, 2n+1\}, |S| \geq 2 \tag{24}$$

$$\sum_{(i,j) \in A(R)} x_{ij} \leq |A(R)| - 1 \qquad \forall R \in \mathcal{R} \tag{25}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in N, j \in N. \tag{26}$$

With formulation (PDPTW2), the DARP can be modeled by simply introducing in set $\mathcal{R}$ the paths violating the ride time constraints.

Constraints (25) can in fact be strengthened into so-called tournament constraints (see, e.g., Ascheuer et al., 2000a) as follows. If $R = (k_1, \ldots, k_r)$ is an infeasible path, then the following inequality is valid:

$$\sum_{i=1}^{r-1} \sum_{j=i+1}^{r} x_{k_i, k_j} \leq |A(R)| - 1. \tag{27}$$

Infeasible path constraints can also be strengthened when they link a node pair $i, n+i$. Consider a path $R = (i, k_1, \ldots, k_r, n+i)$. If R is infeasible because of time windows or ride time constraints (and the triangle inequality holds), then the following inequality is valid (see Cordeau, 2006):

$$x_{i,k_1} + \sum_{h=1}^{r-1} x_{k_h, k_{h+1}} + x_{k_r, n+i} \leq |A(R)| - 2. \tag{28}$$

Finally, if both the path $R = (k_1, \ldots k_r)$ and the reverse path $R' = (k_r, \ldots, k_1)$ are infeasible, then the following symmetric inequality is clearly valid:

$$\sum_{i=1}^{r-1} (x_{k_i, k_{i+1}} + x_{k_{i+1}, k_i}) \leq r - 1. \tag{29}$$

Although formulations (PDPTW1) and (PDPTW2) assume identical vehicles, vehicles of different capacities can be handled through the introduction of dummy requests. Suppose that m vehicles of capacity $Q^1, Q^2, \ldots, Q^m$ are available and let $Q = \max_{1 \leq i \leq m}\{Q^i\}$. One can then define m dummy requests $i = 1, \ldots, m$ with $d_i = d_{n+i} = 0$ and $q_i = -q_{n+i} = Q - Q^i$.

Each dummy pickup node should be reachable only from the origin depot while each dummy delivery node should connect only to the destination depot (both with cost and travel time equal to 0). The arc from a dummy pickup node to a normal pickup node j should have a cost c_{0j} and a travel time t_{0j} while the arc from a normal delivery node $n+j$ to a dummy delivery node should have a cost $c_{n+j,2n+1}$ and a travel time $t_{n+j,2n+1}$. Finally, the corresponding values should be zero for all arcs between dummy pickup and delivery nodes.

Finally note that, as observed by ASCHEUER et al. (2001) in the context of the TSP with time windows, model (PDPTW1) is more flexible than model (PDPTW2) in the sense that it can accomodate a more general objective function involving time and load variables. For example, one could minimize the makespan or a weighted sum of waiting times.

3 Valid Inequalities

We now describe several families of valid inequalities for the PDPTW. These inequalities can be used to strengthen both (PDPTW1) and (PDPTW2). The first two families, subtour elimination constrains and generalized order constraints are borrowed from CORDEAU (2006). The next three families, strengthened capacity constraints, strengthened infeasible path constraints, and fork constraints, are new. The reachability constraints are adapted from existing inequalities for the VRPTW (LYSGAARD, 2004).

Throughout the remainder of the paper, let $x(S) = \sum_{i,j\in S} x_{ij}$ and $x(S:T) = \sum_{i\in S}\sum_{j\in T} x_{ij}$, where $S, T \subseteq N$. For any node subset S, define also $\delta(S) = \delta^+(S) \cup \delta^-(S)$ where $\delta^+(S) = \{(i,j) \in A | i \in S, j \notin S\}$ and $\delta^-(S) = \{(i,j) \in A | i \notin S, j \in S\}$.

3.1 Subtour elimination constraints

Consider the simple subtour elimination constraint $x(S) \leq |S| - 1$ for $S \subseteq P \cup D$. In the case of the PDPTW, this inequality can be lifted in many different ways by taking into account the fact that for each request i, node i must be visited before node $n+i$. For any set $S \subseteq P \cup D$, let $\pi(S) = \{i \in P | n+i \in S\}$ and $\sigma(S) = \{n+i \in D | i \in S\}$ denote the sets of *predecessors* and *successors* of S, respectively. BALAS et al. (1995) have proposed two families of inequalities for the PCATSP which also apply to the PDPTW because each node $i \in P \cup D$ is either the predecessor or the successor of exactly one other node. For $S \subseteq P \cup D$, the following predecessor and successor inequalities are valid for the PDPTW:

$$x(S) + \sum_{i\in S} \sum_{j\in \bar{S}\cap\pi(S)} x_{ij} + \sum_{i\in S\cap\pi(S)} \sum_{j\in \bar{S}\setminus\pi(S)} x_{ij} \leq |S| - 1 \tag{30}$$

$$x(S) + \sum_{i\in \bar{S}\cap\sigma(S)} \sum_{j\in S} x_{ij} + \sum_{i\in \bar{S}\setminus\sigma(S)} \sum_{j\in S\cap\sigma(S)} x_{ij} \leq |S| - 1. \tag{31}$$

As shown by CORDEAU (2006), the D_k^- and D_k^+ inequalities introduced by GRÖTSCHEL and PADBERG (1985) for the asymmetric TSP can also be lifted by taking precedence relationships into account. Let $S = \{i_1, i_2, \ldots, i_k\} \subseteq P \cup D$ be an ordered set of nodes with $k \geq 3$. The following inequalities are then valid for the PDPTW:

$$\sum_{j=1}^{k-1} x_{i_j,i_{j+1}} + x_{i_k,i_1} + 2\sum_{j=3}^{k} x_{i_1,i_j} + \sum_{j=4}^{k}\sum_{l=3}^{j-1} x_{i_j,i_l} + \sum_{h\in\bar{S}\cap\pi(S)} x_{i_1,h} \quad \leq \quad k-1 \tag{32}$$

$$\sum_{j=1}^{k-1} x_{i_j,i_{j+1}} + x_{i_k,i_1} + 2\sum_{j=2}^{k-1} x_{i_j,i_1} + \sum_{j=3}^{k-1}\sum_{l=2}^{j-1} x_{i_j,i_l} + \sum_{h\in\bar{S}\cap\sigma(S)} x_{h,i_1} \quad \leq \quad k-1. \tag{33}$$

3.2 Generalized order constraints

Let $U_1, \ldots, U_s \subset N$ be mutually disjoint subsets and let $i_1, \ldots, i_s \in P$ be requests such that $0, 2n+1 \notin U_l$ and $i_l, n+i_{l+1} \in U_l$ for $l = 1, \ldots, s$ (where $i_{s+1} = i_1$). The following inequality, introduced by Ruland and Rodin (1997), is also valid for the PDPTW:

$$\sum_{l=1}^{s} x(U_l) \leq \sum_{l=1}^{s} |U_l| - s - 1. \tag{34}$$

Similar inequalities, called *precedence cycle breaking inequalities*, have also been proposed by Balas et al. (1995) for the PCATSP. In the case of a directed formulation, Cordeau (2006) has shown that generalized order constraints can be lifted in two different ways as follows:

$$\sum_{l=1}^{s} x(U_l) + \sum_{l=2}^{s-1} x_{i_1,i_l} + \sum_{l=3}^{s} x_{i_1,n+i_l} \leq \sum_{l=1}^{s} |U_l| - s - 1 \tag{35}$$

$$\sum_{l=1}^{s} x(U_l) + \sum_{l=2}^{s-2} x_{n+i_1,i_l} + \sum_{l=2}^{s-1} x_{n+i_1,n+i_l} \leq \sum_{l=1}^{s} |U_l| - s - 1. \tag{36}$$

3.3 Strengthened capacity constraints

Capacity constraints can be strengthened by considering node pairs $(k, n+k)$ such that the pickup node k is visited before entering set S while the delivery node $n+k$ is visited after leaving this set. In this case, the number of vehicles visiting set S increases to accommodate the demand of all such node pairs. This yields the following result.

Proposition 1. Let $S, T \subset P \cup D$ be two disjoint sets such that $q(S) > 0$. Also define $U = \pi(T) \setminus (S \cup T)$. The following inequality is then valid for the PDPTW:

$$x(S) + x(T) + x(S:T) \leq |S| + |T| - \left\lceil \frac{q(S)+q(U)}{Q} \right\rceil. \tag{37}$$

Proof. Because $q(S) > 0$ and $q(U) \geq 0$, $x(\delta^+(S)) \geq \lceil q(S)/Q \rceil$ and $x(\delta^-(T)) \geq \lceil q(U)/Q \rceil$. If a path uses an arc from the set $(S:T)$ and reaches a node $n+k \in T$ with $k \in U$, without leaving set T, then node k must have been visited by that path before entering set S. Hence,

$$x(\delta^+(S)) + x(\delta^-(T)) - x(S:T) \;\;\geq\;\; \left\lceil \frac{q(S)+q(U)}{Q} \right\rceil .$$

Because $x(S) + x(\delta^-(S)) = x(S) + x(\delta^+(S)) = |S|$, this is equivalent to

$$|S| - x(S) + |T| - x(T) - x(S:T) \;\;\geq\;\; \left\lceil \frac{q(S)+q(U)}{Q} \right\rceil ,$$

which yields the desired result after properly rearranging the terms.□

One may observe that inequalities (37) constitute a strengthening of the rounded capacity inequalities. From the inequalities

$$x(s) \leq |S| - \left\lceil \frac{q(S)}{Q} \right\rceil$$

and

$$x(T) + X(S:T) \leq x(T) + \delta^-(T) = |T|,$$

one obtains

$$x(S) + x(T) + x(S:T) \leq |S| + |T| - \left\lceil \frac{q(S)}{Q} \right\rceil ,$$

which is weaker than (37) if $q(U) > 0$.

Figure 1 depicts an example for which at most two arcs can be used if $q_i = q_j = q_k = q_l = 1$ and the vehicle capacity is $Q = 2$. Arcs in the figure are those on the left-hand-side of (37).

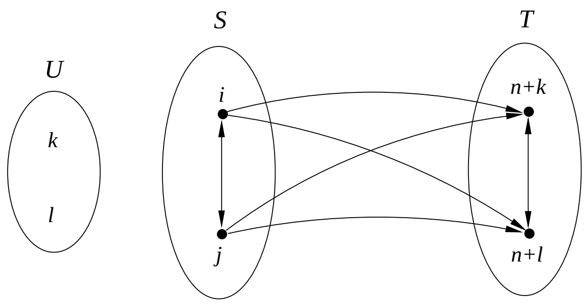


Figure 1: Strengthened capacity constraint where $S = \{i, j\}$, $T = \{n+k, n+l\}$ and $U = \pi(T) \setminus (S \cup T) = \{k, l\}$.

3.4 Strengthened infeasible path constraints

Paths that satisfy time windows can sometimes be eliminated by taking precedence relationships into account. Consider for instance the path $R = (i, n+j, k)$. Obviously, node j must be visited before R, while nodes $n+i$ and $n+k$ must be visited after R. Hence, if both

$(j, i, n+j, k, n+i, n+k)$ and $(j, i, n+j, k, n+k, n+i)$ are infeasible, then R cannot belong to a feasible solution. More generally, let $\phi(S)$ denote the set of all permutations of nodes in S. For any path R, denote by $N(R) \subseteq N$ the set of nodes visited by that path. If R is a feasible path in G but (ϕ_p, R, ϕ_d) is infeasible for all $\phi_p \in \phi(\pi(R) \setminus N(R))$ and $\phi_d \in \phi(\sigma(R) \setminus N(R))$ then R cannot belong to a feasible solution and it can thus be eliminated by (27).

3.5 Fork constraints

Infeasible paths can also be eliminated in a different way by considering groups of infeasible paths sharing some common arcs. For instance, if the path $R = (k_1, \dots, k_r)$ is feasible, but the path (i, R, j) is infeasible for every $i \in S$ and $j \in T$ with $S, T \subset N$, then the following inequality is clearly valid:

$$\sum_{i \in S} x_{i,k_1} + \sum_{h=1}^{r-1} x_{k_h,k_{h+1}} + \sum_{j \in T} x_{k_r,j} \leq r. \tag{38}$$

This inequality can be strengthened by associating to each intermediate node $k_2, \dots, k_{r-1}$ a set of nodes leading to infeasible paths. This results in the following *outfork inequality*.

Proposition 2. Let $R = (k_1, \dots, k_r)$ be a feasible path in G and $S, T_1, \dots, T_r \subset (P \cup D) \setminus N(R)$ be subsets such that for any integer $h \leq r$ and any node pair $i \in S, j \in T_h$, the path $(i, k_1, \dots, k_h, j)$ is infeasible. The following inequality is then valid for the PDPTW:

$$\sum_{i \in S} x_{i,k_1} + \sum_{h=1}^{r-1} x_{k_h,k_{h+1}} + \sum_{h=1}^{r} \sum_{j \in T_h} x_{k_h,j} \leq r. \tag{39}$$

Proof. Assume that the inequality is violated in a feasible integer solution. Then, among the arcs belonging to the inequality, $r+1$ must have been selected. Because of the degree constraints, there must be one arc from S to k_1, one outgoing arc from each node $k_1, \dots, k_{r-1}$, and one arc from k_r to T_r. As a result, the path originating in S reaches one of the nodes in the sets $T_h, 1 \leq h \leq r$, and must thus be infeasible.$\square$

The outfork inequality is illustrated in Figure 2 for the case $r = 3$. Similar inequalities, called *infork inequalities* and illustrated in Figure 3 for the case $r = 3$, are obtained by reversing the orientation of the arcs reaching path R. These lead to the following proposition.

Proposition 3. Let $R = (k_1, \dots, k_r)$ be a feasible path in G and $S_1, \dots, S_r, T \subset (P \cup D) \setminus N(R)$ be subsets such that for any integer $h \leq r$ and any node pair $i \in S_h, j \in T$, the path $(i, k_h, \dots, k_r, j)$ is infeasible. The following inequality is then valid for the PDPTW:

$$\sum_{h=1}^{r} \sum_{i \in S_h} x_{i,k_h} + \sum_{h=1}^{r-1} x_{k_h,k_{h+1}} + \sum_{j \in T} x_{k_r,j} \leq r. \tag{40}$$

It is worth pointing out that fork constraints can be used in any routing problem where the concept of infeasible paths is well defined, for instance the vehicle routing problem with time windows.

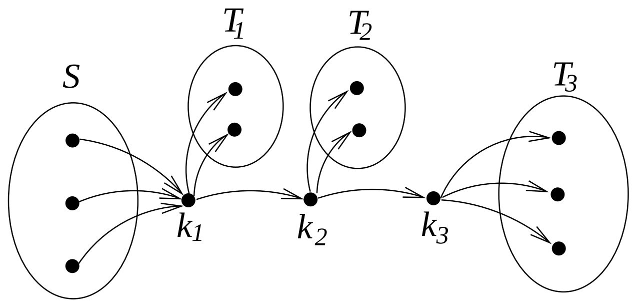


Figure 2: Outfork constraint with $r = 3$

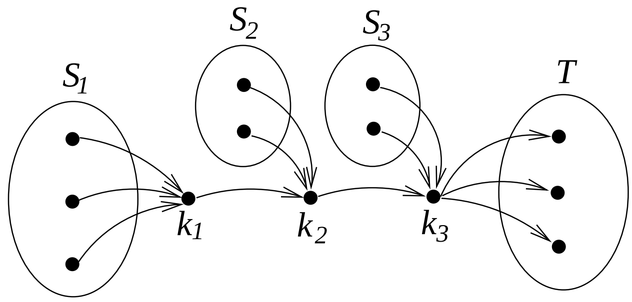


Figure 3: Infork constraint with $r = 3$

3.6 Reachability constraints

For any node $i \in N$, let $A_i^- \subset A$ be the minimum arc set such that any feasible path from the origin depot 0 to node i uses only arcs from A_i^-. Let also A_i^+ be the minimum arc set such that any feasible path from i to the destination depot $2n+1$ uses only arcs in A_i^+. Consider a node set T such that each node in T must be visited by a different vehicle. This set is said to be *conflicting*. For any conflicting node set T, define the *reaching arc set* $A_T^- = \cup_{i \in T} A_i^-$ and the *reachable arc set* $A_T^+ = \cup_{i \in T} A_i^+$. For any node set $S \subseteq P \cup D$ and any conflicting node set $T \subseteq S$, the following two valid inequalities were introduced by Lysgaard (2004) for the VRP with time windows:

$$x(\delta^-(S) \cap A_T^-) \geq |T| \tag{41}$$
$$x(\delta^+(S) \cap A_T^+) \geq |T|. \tag{42}$$

These inequalities are obviously also valid for the PDPTW. In this problem, however, nodes can be conflicting not only because of time windows but also because of the precedence relationships and the capacity constraints. In the case of the DARP, the ride time constraints should also be taken into account when checking whether a pair of requests is conflicting.

4 Branch-and-Cut Algorithms

We have implemented two branch-and-cut algorithms for the PDPTW: one with formulation (PDPTW1) and one with formulation (PDPTW2). In both algorithms, an attempt is made to generate violated valid inequalities at each node of the search tree. With formulation (PDPTW1), precedence inequalities (4) must be generated to ensure feasibility. With formulation (PDPTW2), feasibility is ensured by generating not only the precedence inequalities (23), but also the capacity inequalities (24) and infeasible path inequalities (25). In both formulations, the additional inequalities described in the previous section can be used to improve the LP relaxation obtained at each node of the branch-and-bound tree. In addition, inequalities (24) and (25) can be used to strengthen formulation (PDPTW1) although these are not required to ensure feasiblity.

Taking into account the precedence relationships, time windows and ride time constraints, several arc elimination rules can be used in a preprocessing step to reduce the size of the problem. In addition, time windows can often be tightened. Details on these preprocessing steps can be found in the papers of Dumas et al. (1991) and Cordeau (2006).

In both branch-and-cut algorithms, the LP relaxations are solved by the simplex algorithm. Branching is performed on the x_{ij} variables by choosing, at each node of the enumeration tree, the variable whose value is the farthest from the nearest integer. The search is performed by applying the best-bound strategy. Prior to solving the problem, an upper bound is computed by using either the adaptive large neighbourhood search algorithm of Ropke and Pisinger (2004) for the PDPTW or the tabu search heuristic of Cordeau and Laporte (2003) for the DARP.

We now describe the separation procedures used to generate the precedence, capacity and infeasible path inequalities. We then describe procedures for the additional inequalities introduced in Section 3.

4.1 Precedence constraints

Violated precedence constraints (4) and (23) can be identified in polynomial time by solving a series of maximum flow problems: for each request i, one can compute the maximum flow from nodes i and $2n+1$ to nodes 0 and $n+i$ in G, with arc capacities given by the values of the x_{ij} variables. If the value of this flow is less than 1, then a precedence constraint is violated for a set S such that $0, n+i \in S$ and $i, 2n+1 \notin S$. The set S corresponds to one of the shores of the corresponding minimum cut. We have implemented this procedure by using the Ford-Fulkerson algorithm described by Cormen et al. (1990).

4.2 Capacity constraints

Two heuristics are used for the identification of violated capacity constraints. The first one is a randomized construction heuristic which starts from a given node $i \in P \cup D$ and gradually adds nodes to S by considering, at each iteration, the nodes connected to S with some flow. The choice of the node being added to S from the set of potential nodes is done randomly (with each node having a probability of being selected proportional to the flow on the corresponding arc). The procedure is repeated several times for each start node. If a capacity constraint is violated in an integer solution, the violation will clearly be detected by this procedure since it will add, at each iteration, the only node connected to the previously added node. At some point during the process, the set S will thus satisfy $q(S) > Q$.

The second heuristic is a simple tabu search heuristic described by CORDEAU (2006) and inspired by that originally proposed by AUGERAT et al. (1999). This heuristic starts with either a random subset $S \subseteq P$ or a random subset $S \subseteq D$. At each iteration, a node is either removed or added to S so as to minimize the value of $x(\delta(S))$ while satisfying $q(S) > Q$.

4.3 Subtour elimination constraints

It is well known that the separation problem for subtour elimination constraints is solvable in polynomial time by computing the maximum flow between each node i and all other nodes $j \in N \setminus \{i\}$. This procedure, however, does not take into account the various liftings proposed in inequalities (30)-(33). Hence, we resort here to a simple tabu search heuristic very similar to the one used for capacity constraints and also described in more detail by CORDEAU (2006).

4.4 Generalized order constraints

We use two simple heuristics for the lifted generalized order constraints (35) and (36). These heuristics consider the case where $m = 3$ and $|U_1| = |U_2| = |U_3| = 2$. The first heuristic identifies, for each user i, a user j maximizing $x_{i,n+j} + x_{n+j,i} + x_{ij}$. It then finds a user k such that the left-hand side of (35) is maximized. The second heuristic identifies, for each user i, a user j maximizing $x_{i,n+j} + x_{n+j,i} + x_{n+i,n+j}$ and then a user k maximizing the left-hand side of (36).

4.5 Strengthened capacity constraints

To identify sets S and T for which the strengthened capacity constraint is violated, we use a construction heuristic similar to that used for the capacity constraints. This procedure starts from a set S containing a single pickup node and gradually augments this set by adding one node at a time. Before augmenting the set, the procedure determines

$$b_p \in \arg\max_{i \in P \setminus S} \{x(S:i) + x(i:S)\}$$

and

$$b_d \in \arg\max_{i \in D \setminus S} \{x(S:i) + x(i:S)\}$$

which we consider to be the best pickup (resp. delivery) node to add to the set. We prefer to add a pickup node to S in order to increase $q(S)$ on the right-hand side of inequality (37). Node b_d is only added if $x(S:b_p)+x(b_p:S) < x(S:b_d)+x(b_d:S)$, $q(S\cup\{b_d\})>0$ and either $x(S:b_d)+x(b_d:S)\geq 1$ or $x(S:i)+x(i:S)=0$ for all $i\in P\setminus S$. Each time a node is added to S, the set T is reconstructed by using a similar construction heuristic where the roles of pickups and deliveries are interchanged. Only nodes from $N\setminus S$ are added to T.

At the root node of the search tree we use a modified version of this heuristic where a random perturbation $\epsilon\sim[0,0.5]$ is added to the evaluation of $x(S:i)+x(i:S)$ and the heuristic is restarted several times from each pickup node.

4.6 Strengthened infeasible path constraints

To identify infeasible paths violating constraints (25), we use an enumerative procedure similar to that of Ascheuer et al. (2001). In this procedure, every node $i\in P\cup D$ is in turn considered as a start node from which a tree of paths with positive flow is constructed. Each path is extended as long as a violation along this path is still possible (i.e., as long as the total flow on the arcs in path R is strictly greater than $|A(R)|-1$ and the path has not reached node $2n+1$). Each time an infeasible path is identified, the corresponding tournament constraint (27) is generated.

A very similar procedure is used to identify violated strengthened infeasible path constraints for the DARP. In this case, however, each node $i\in P$ is considered as a start node and the extension of a path also stops if it reaches node $n+i$, at which point it is checked for feasibility with respect to the time windows and the ride time constraint for user i.

4.7 Fork constraints

A partial enumeration procedure is used to separate the fork constraints with $r=1$. This procedure first enumerates the set H of all infeasible paths containing three nodes. To identify violated outfork constraints, it starts from an arc (i,j) and constructs the set S by identifying all paths of the form (h,i,j) belonging to H. Finally, the set T_1 is constructed by identifying all nodes k such that $(h,i,k)\in H$ for every node $h\in S$. This procedure is repeated for every arc (i,j) for which $x_{ij}>0$ in the current solution. To identify violated infork constraints, a similar procedure starts from an arc (i,j) and constructs a set T containing all nodes k such that $(i,j,k)\in H$. The set S_1 is then constructed by identifying all nodes h such that $(h,j,k)\in H$ for every node $k\in T$.

For $r\geq 2$ a different heuristic is used. The heuristic iteratively uses every node $k_0\in P\cup D$ as a seed node. From k_0, feasible paths $(k_0,k_1,\ldots,k_l)$ are gradually constructed by extending existing paths along arcs with positive flow. For every path, one then checks if a violated fork constraint can be found with the path as a backbone. First, the set T is constructed such that $(k_0,k_1,\ldots,k_l,j)$ is infeasible for all $j\in T$. Then, the set $S\ni k_0$ is constructed such that all paths $(i,k_1,\ldots,k_l,j), i\in S, j\in T$ are infeasible. The two sets S and T and the path $(k_1, \ldots, k_l)$ define a simple fork inequality (38). If this inequality is not violated, the procedure attempts to lift it into an outfork or an infork inequality. To lift the inequality into an outfork inequality, one adds as many nodes as possible to the sets $T_1, \ldots, T_l$. A similar approach is used to lift the inequality into an infork. In order to keep running times low, only paths containing at most six nodes are considered. Checking whether a path is infeasible can be time consuming as many permutations have to be examined as described in section 3.4. To alleviate this problem the feasibility of a path is only checked once, and the result of the query is stored in a hash table from which it can be quickly retrieved.

4.8 Reachability constraints

Our procedure first computes, for each node $i \in P \cup D$, the sets A_i^+ and A_i^-. When doing this, precedence relationships must be taken into account. For example, when checking whether an arc $(i, n+j)$ belongs to the set A_{n+k}^-, one must check the existence of a path containing this arc and such that k is visited before $n+k$, j is visited before $n+j$, and $n+i$ is visited after i in this path. The procedure then identifies, by complete enumeration, all sets of conflicting requests with a cardinality smaller than or equal to a given threshold. Each set of conflicting requests gives rise to several sets of conflicting nodes. For a set of k conflicting requests, 2^k sets of conflicting nodes exist. When k is greater than a parameter τ we do not generate all conflicting node sets, but only those two consisting of either the pickups or the deliveries of the conflicting requests. For a fractional solution, one then considers each conflicting node set T and solves a maximum flow problem between the node 0 and the set T by considering only the arcs in A_T^-. If the capacity of the corresponding minimum cut is smaller than $|T|$, then a violation of a reachability cut has been found. The same is done by considering A_T^+ and solving a maximum flow problem between set T and the destination depot $2n+1$.

5 Computational Experiments

The two branch-and-cut algorithms were implemented in C++ by using ILOG Concert 1.3 and CPLEX 9.0. All experiments were performed on a AMD Opteron 250 computer (2.4GHz). Several sets of instances for the PDPTW and the DARP were used for testing. All instances are available on `http://www.hec.ca/chairedistributique/data`.

5.1 Results for the PDPTW

We first generated some PDPTW instances as suggested by Savelsbergh and Sol (1998). In these instances, the coordinates of each pickup and delivery location are chosen randomly according to a uniform distribution over the $[0, 200] \times [0, 200]$ square. The load q_i of request i is selected randomly from the interval $[5, Q]$, where Q is the vehicle capacity. A planning horizon of length $H = 600$ is considered and each time window has width W. The time windows for request i are constructed by first randomly selecting e_i in the interval $[0, H - t_{i,n+i}]$ and then setting $l_i = e_i + W$, $e_{n+i} = e_i + t_{i,n+i}$ and $l_{n+i} = e_{n+i} + W$. In all instances, the primary objective consists of minimizing the number of vehicles, and a fixed cost of 10^4 is thus imposed on each outgoing arc from the depot. Four classes of instances are obtained by varying the values of Q and W, as indicated in the following table.

Table 1: Characteristics of the Savelsbergh and Sol PDPTW instances

Class	Q	W
A	15	60
B	20	60
C	15	120
D	20	120

In the test instances generated by Savelsbergh and Sol (1998), each vehicle has a different depot whose location is also chosen randomly over the $[0, 200] \times [0, 200]$ square. Because our formulations cannot handle multiple depots directly, we have instead used a single depot located in the middle of the square.

As is apparent from the results reported by Savelsbergh and Sol (1998), using the $[0, 200] \times [0, 200]$ square with $H = 600$ yields instances in which it is difficult to serve more than two or three requests in the same route. In addition, the long travel times make it difficult to stop at an intermediate location between the pickup of a request and its delivery. As a result, all instances generated in this way could be solved at the root node by our algorithms. To obtain harder instances, we have decreased the size of the square from which the locations are chosen. By choosing coordinates from the set $[0, 50] \times [0, 50]$, travel times become smaller and it is then possible to serve more requests in each route. Furthermore, it becomes easier to produce a sequence of several successive pickups followed by the corresponding deliveries. In each of the four problem classes, we have generated ten instances by considering values of n between 30 and 75. The name of each instance (e.g., A50) indicates the class to which it belongs and the number of requests it contains.

We first present in Table 2 the solution values and computing times (in minutes) of the Ropke and Pisinger (2004) adaptive large neighbourhood search heuristic for the PDPTW. The table also shows the graph density after preprocesing, calculated as

$$100 \times \frac{\text{number of arcs}}{(2n+2)^2}$$

To evaluate the strength of formulations (PDPTW1) and (PDPTW2), we have first solved the LP relaxation of both formulations by considering the minimal sets of inequalities required for feasibility. Hence, violated precedence constraints were generated for (PDPTW1), while for (PDPTW2) we have also generated violated capacity constraints and infeasible path constraints. These results are reported in Table 3. For each instance, we indicate in columns LP1 and LP2 the value of the lower bound computed at the root node as a percentage of

Table 2: Solution values and computing times for the heuristic

Instance	UB heuristic	Time heuristic	Graph density (%)
A30	51,317.40	0.45	14.9
A35	51,343.53	0.55	18.4
A40	61,609.44	0.69	15.9
A45	61,693.01	0.83	25.2
A50	71,932.03	0.98	17.5
A55	82,185.31	1.07	19.2
A60	92,366.70	1.28	16.9
A65	82,331.12	1.46	17.3
A70	112,458.28	1.64	16.3
A75	92,529.42	1.88	20.6
B30	51,193.62	0.47	21.9
B35	61,400.07	0.54	20.4
B40	51,421.35	0.74	20.4
B45	61,787.28	0.82	21.5
B50	71,889.75	0.98	20.9
B55	82,080.73	1.07	18.6
B60	102,323.77	1.23	14.7
B65	82,623.98	1.42	19.7
B70	92,647.75	1.68	18.9
B75	92,476.30	1.88	20.1
C30	51,145.18	0.47	14.4
C35	51,235.64	0.57	17.6
C40	61,473.91	0.72	18.6
C45	81,408.89	0.83	21.1
C50	61,936.27	1.06	20.1
C55	61,930.55	1.19	20.9
C60	72,104.00	1.38	18.0
C65	82,326.62	1.50	24.0
C70	92,613.68	1.70	19.0
C75	92,711.74	1.88	21.4
D30	61,040.10	0.46	22.9
D35	71,308.04	0.56	25.6
D40	61,531.68	0.72	25.5
D45	81,601.63	0.80	17.7
D50	71,761.23	1.00	20.9
D55	72,051.95	1.15	21.7
D60	82,308.08	1.31	18.0
D65	82,200.77	1.50	24.9
D70	82,631.56	1.70	19.2
D75	92,970.84	1.83	21.6

the upper bound indicated in the rightmost column of the table. This upper bound is either the optimal value of the problem, if the instance could be solved to optimality, or an upper bound computed by a heuristic, otherwise. One can see that for most instances (PDPTW2) provides a tighter lower bound, with an average of 72.33 for (PDPTW2) compared to 69.90 for (PDPTW1). In Tables 3 and 4, the number of vehicles in the solution is equal to $\lfloor U/10^4 \rfloor$ where U is the upper bound.

To measure the strength of each type of inequality introduced in Section 3, we then solved the LP relaxation of (PDPTW2) by separately considering each type of inequality: subtour elimination constraints (SEC), strengthened capacity constraints (SCC), generalized order constraints (GOC), fork constraints (FC) and reachability constraints (RC). Finally, column "Full" reports the lower bound obtained with (PDPTW2) when considering all families of valid inequalities. Again, all lower bounds are expressed as a percentage of the upper bound reported in the last column of the table. These results show that fork constraints and reachability constraints have the largest impact, with all other types of inequalities playing only a minor role in the improvement of the lower bound. It is worth pointing out that for some instances (e.g., B55), the lower bound obtained with one type of inequality is sometimes worse than that obtained with just the basic formulation (column LP2). This is explained by the fact that we use a heuristic separation procedure for capacity constraints, which may lead to the generation of a different set of inequalities.

In Table 4, we report the results obtained by considering both formulations with all types of valid inequalities. For each instance that was solved to optimality, we indicate the CPU time in minutes (including the preprocessing time) needed to prove optimality, the number of nodes explored in the search tree and the total number of cuts generated during the search. When an instance could not be solved to optimality within the maximum CPU time (two hours), we report the value of the current lower bound at the end of the computation (i.e., the lower bound associated with the best pending node). These results show that formulation (PDPTW2) provides a slightly better performance: it solved five more instances to optimality and for those instances that were solved by both formulations, (PDPTW2) required on average less CPU time, fewer nodes and fewer cuts. Finally, when neither model could reach an optimal solution, the latter usually provided a higher lower bound.

5.2 Results for the DARP

We have then tested our approach on two sets of randomly generated Euclidean DARP instances comprising up to 96 requests. These instances have narrow time windows of 15 minutes. In the first set ('a' instances), $q_i = 1$ for every request i and the vehicle capacity is $Q = 3$. In the second set ('b' instances), q_i belongs to the interval $[1, 6]$ and $Q = 6$. These data are described in detail in Cordeau (2006) and their main characteristics are summarized in Table 5. In this table, columns $|K|$ and H indicate, respectively, the number of available vehicles and the length of the planning horizon in which time windows are generated. The constraint on the number of vehicles is easily imposed in our formulations as a bound on the total outgoing flow from the origin depot. Finally, L denotes the maximum ride time.

We present in Table 6 the solution values and computing times for the Cordeau and

Table 3: Lower bounds obtained in the root node as a percentage of the upper bound

	LP1	LP2	SEC	SCC	GOC	FC	RC	FULL	U. Bound
A30	99.95	99.98	99.98	99.98	99.98	100.00	99.99	100.00	51,317.40
A35	99.68	99.84	99.85	99.86	99.84	100.00	99.99	100.00	51,343.53
A40	97.42	99.85	99.85	99.86	99.85	100.00	100.00	100.00	61,609.44
A45	83.21	83.35	83.35	83.39	83.35	83.90	83.83	83.99	61,693.01
A50	78.09	72.20	72.20	74.53	72.20	93.13	99.99	100.00	71,932.03
A55	71.49	88.50	88.50	88.52	88.17	93.85	99.91	99.95	82,185.31
A60	83.70	92.19	92.19	92.21	92.19	100.00	100.00	100.00	92,366.70
A65	89.27	89.23	89.23	89.24	89.23	99.99	99.97	100.00	82,331.12
A70	82.03	91.00	91.00	91.01	91.00	93.93	93.98	95.60	112,458.28
A75	57.29	70.27	70.27	70.30	70.27	78.44	99.89	99.97	92,525.46
B30	85.21	84.36	84.36	84.36	84.36	99.99	99.99	99.99	51,193.62
B35	67.35	70.74	70.74	72.89	70.74	89.24	91.92	91.93	61,400.07
B40	64.97	65.45	65.45	65.48	65.45	83.55	80.86	85.72	51,421.35
B45	67.29	67.51	67.51	69.73	67.51	99.96	99.92	99.97	61,787.28
B50	51.32	66.52	66.52	66.53	66.52	87.90	99.95	99.98	71,889.75
B55	63.37	58.88	58.89	59.85	58.89	99.98	99.97	99.99	82,080.73
B60	80.37	80.43	80.43	80.43	80.43	90.58	99.99	100.00	102,323.77
B65	85.88	75.39	75.39	75.40	75.39	94.41	99.89	99.93	82,617.22
B70	61.03	67.32	67.32	67.34	67.32	94.57	99.92	99.96	92,641.67
B75	56.32	58.66	60.10	59.05	58.93	85.46	89.23	89.35	92,476.30
C30	90.17	90.28	90.28	90.28	90.28	100.00	99.99	100.00	51,145.18
C35	80.24	80.33	80.33	80.35	80.33	80.72	99.94	99.98	51,235.64
C40	67.20	67.32	67.33	67.34	67.32	83.80	83.78	83.83	61,473.91
C45	50.74	75.51	75.58	75.61	75.60	87.76	99.96	100.00	81,405.96
C50	99.40	99.52	99.52	99.53	99.52	99.92	99.86	99.94	61,933.09
C55	67.04	67.20	67.21	67.23	67.20	91.90	99.81	99.92	61,930.55
C60	57.85	68.07	68.07	68.10	68.07	99.86	99.80	99.89	72,100.68
C65	53.96	54.84	54.84	54.86	54.84	76.57	99.70	99.79	82,326.62
C70	56.35	56.47	56.48	56.48	56.48	84.96	89.11	89.22	92,613.68
C75	56.17	67.03	67.04	67.06	67.03	78.33	99.71	99.82	92,711.74
D30	64.68	67.16	67.16	67.18	67.15	88.45	99.95	99.99	61,040.10
D35	46.34	47.20	47.19	47.21	47.20	58.04	99.86	99.93	71,308.04
D40	67.00	67.11	67.12	67.15	67.11	99.86	99.79	99.87	61,531.68
D45	87.76	87.56	87.56	87.56	87.56	99.98	99.98	99.99	81,601.52
D50	54.60	57.99	57.99	58.03	57.99	86.14	99.92	99.99	71,761.23
D55	52.59	57.95	57.95	58.00	57.96	86.06	99.80	99.90	72,051.95
D60	75.36	75.40	75.40	75.41	75.40	99.97	99.91	99.98	82,306.47
D65	49.05	38.71	38.72	38.78	38.73	93.77	99.72	99.85	82,200.77
D70	55.52	51.12	51.13	51.14	51.12	79.38	99.73	99.83	82,631.56
D75	38.78	34.84	34.84	34.89	37.51	63.49	99.62	99.76	92,970.84
avg.	69.90	72.33	72.37	72.55	72.40	90.20	97.73	97.95	

Laporte (2003) tabu search heuristic for the DARP. The table also shows the graph density after preprocesing.

Table 4: Computational results for PDPTW instances

		(PDPTW1)				(PDPTW2)			
Instance	U. Bound	Time	Nodes	Cuts	L. Bound	Time	Nodes	Cuts	L. Bound
A30	51,317.40	0.05	0	78		0.05	0	140	
A35	51,343.53	0.11	0	407		0.10	0	602	
A40	61,609.44	0.16	0	382		0.16	0	510	
A45	61,693.01				51,824.39				51,842.89
A50	71,932.03	0.52	0	838		0.41	0	988	
A55	82,185.31	18.30	855	3877		4.75	91	3704	
A60	92,366.70	0.86	0	766		0.81	0	1071	
A65	82,331.12	2.15	2	1325		1.75	0	1423	
A70	112,458.28				109,244.57	4.74	11	2492	
A75	92,525.46				92,503.50	36.82	438	6414	
B30	51,193.62	0.10	2	426		0.10	7	549	
B35	61,400.07	0.18	2	462		0.15	2	743	
B40	51,421.35	1.26	70	1638		0.52	17	1134	
B45	61,787.28	1.53	98	1636		0.87	31	1814	
B50	71,889.75	6.19	603	2134		1.32	8	2441	
B55	82,080.73	1.06	2	1138		1.07	2	1684	
B60	102,323.77	2.26	4	1891		2.17	6	1975	
B65	82,617.22				82,573.37	77.00	1884	10505	
B70	92,641.67	109.08	3012	5777		13.97	158	4578	
B75	92,476.30				82,649.55				84,105.71
C30	51,145.18	0.06	0	103		0.06	0	158	
C35	51,235.64	0.32	5	920		0.26	6	1059	
C40	61,473.91				51,565.97				51,628.73
C45	81,405.96	0.94	4	1515		0.69	6	1994	
C50	61,933.09	40.64	1541	5637		7.49	228	3989	
C55	61,930.55	96.36	2529	8311		19.32	345	6188	
C60	72,100.68				72,053.40	76.54	3294	10498	
C65	82,326.62				82,159.87				82,163.46
C70	92,613.68				82,666.21				86,645.63
C75	92,711.74				92,555.04				92,554.26
D30	61,040.10	0.30	12	896		0.23	18	1166	
D35	71,308.04				71,299.29	33.07	3155	8618	
D40	61,531.68				61,463.09				61,493.63
D45	81,601.52	1.16	34	1167		0.82	27	1270	
D50	71,761.23	4.22	204	2100		1.61	16	2216	
D55	72,051.95				72,001.09				72,034.13
D60	82,306.47	5.36	144	2098		3.25	51	2589	
D65	82,200.77				82,091.05				82,122.81
D70	82,631.56				82,493.27				82,514.97
D75	92,970.84				92,751.85				92,751.63

Table 5: Characteristics of DARP instances

Instance	$\lvert K\rvert$	n	H	Q	L	Instance	$\lvert K\rvert$	n	H	Q	L
a2-16	2	16	480	3	30	b2-16	2	16	480	6	45
a2-20	2	20	600	3	30	b2-20	2	20	600	6	45
a2-24	2	24	720	3	30	b2-24	2	24	720	6	45
a3-24	3	24	480	3	30	b3-24	3	24	480	6	45
a3-30	3	30	600	3	30	b3-30	3	30	600	6	45
a3-36	3	36	720	3	30	b3-36	3	36	720	6	45
a4-32	4	32	480	3	30	b4-32	4	32	480	6	45
a4-40	4	40	600	3	30	b4-40	4	40	600	6	45
a4-48	4	48	720	3	30	b4-48	4	48	720	6	45
a5-40	5	40	480	3	30	b5-40	5	40	480	6	45
a5-50	5	50	600	3	30	b5-50	5	50	600	6	45
a5-60	5	60	720	3	30	b5-60	5	60	720	6	45
a6-48	6	48	480	3	30	b6-48	6	48	480	6	45
a6-60	6	60	600	3	30	b6-60	6	60	600	6	45
a6-72	6	72	720	3	30	b6-72	6	72	720	6	45
a7-56	7	56	480	3	30	b7-56	7	56	480	6	45
a7-70	7	70	600	3	30	b7-70	7	70	600	6	45
a7-84	7	84	720	3	30	b7-84	7	84	720	6	45
a8-64	8	64	480	3	30	b8-64	8	64	480	6	45
a8-80	8	80	600	3	30	b8-80	8	80	600	6	45
a8-96	8	96	720	3	30	b8-96	8	96	720	6	45

Tables 7 and 8 show the strength of the lower bounds obtained with the different types of valid inequalities. These tables can be interpreted in the same way as Table 3. This time, however, we also indicate in column LP0 the lower bound obtained with the three-index formulation of Cordeau (2006). Again, formulation (PDPTW2) provides better bounds than (PDPTW1) while fork constraints and reachability constraints are the most useful. One can also see that both (PDPTW1) and (PDPTW2) do much better than the three-index formulation in terms of the initial lower bound.

Finally, Tables 9 and 10 report the computational statistics collected when solving each instance to optimality with both (PDPTW1) and (PDPTW2), again using all types of inequalities. In column (DARP), we also indicate comparable statistics for the three-index DARP fomulation of Cordeau (2006). For the latter formulation, only a small subset of all instances could be solved to optimality. As in Table 4, one can see that formulation (PDPTW2) usually requires less computation time and a smaller number of branch-and-bound nodes than (PDPTW1). The largest CPU time for (PDPTW1) is 1210.56 minutes compared to 120.09 minutes for (PDPTW2). Comparisons with the three-index DARP formulation show that the latter is totally dominated by the two new formulations. For example, instance b4-32 required more than one hour of CPU time with the DARP formulation (and 44877 branch-and-cut nodes) while it was solved in the root node with both (PDPTW1) and (PDPTW2). This dramatic improvement results from the improved lower bound provided by the tighter (PDPTW1) and (PDPTW2) formulations, and from the new

Table 6: Solution values and computing times (in minutes) for the DARP instances

Instance	U. Bound heuristic	Time heuristic	Graph density (%)	Instance	U. Bound heuristic	Time heuristic	Graph density (%)
a2-16	294.25	0.05	31.2	b2-16	309.61	0.05	28.1
a2-20	344.83	0.12	32.1	b2-20	334.93	0.10	18.9
a2-24	431.12	0.21	32.6	b2-24	445.11	0.21	23.2
a3-24	344.83	0.12	35.2	b3-24	394.57	0.11	23.3
a3-30	496.52	0.24	31.6	b3-30	536.04	0.24	23.5
a3-36	600.75	0.48	31.2	b3-36	611.79	0.46	21.8
a4-32	486.57	0.20	32.9	b4-32	500.92	0.17	20.8
a4-40	571.07	0.38	32.3	b4-40	662.91	0.38	20.2
a4-48	680.99	0.75	34.8	b4-48	685.46	0.77	27.0
a5-40	507.59	0.28	33.5	b5-40	619.09	0.26	27.2
a5-50	699.86	0.58	34.8	b5-50	777.20	0.55	23.5
a5-60	825.57	1.20	32.8	b5-60	923.07	1.08	27.6
a6-48	618.00	0.37	34.2	b6-48	727.06	0.31	21.7
a6-60	847.19	0.75	33.1	b6-60	888.28	0.77	23.3
a6-72	946.41	1.50	33.8	b6-72	1007.99	1.44	24.3
a7-56	745.08	0.52	32.3	b7-56	844.54	0.45	24.0
a7-70	936.96	1.00	33.4	b7-70	939.10	0.89	21.7
a7-84	1069.77	1.87	33.6	b7-84	1255.10	1.78	25.3
a8-64	770.52	0.69	33.9	b8-64	865.65	0.54	23.7
a8-80	992.52	1.22	35.0	b8-80	1085.91	1.33	21.3
a8-96	1289.59	2.55	33.5	b8-96	1236.42	2.36	26.0

inequalities introduced in this paper.

6 Conclusion

By using appropriate inequalities, we have introduced two new formulations for the PDPTW which do not require the use of a vehicle index to impose pairing and precedence constraints, as is the case in three-index formulations. In addition to adapting infeasible path constraints and reachability constraints to take advantage of the structure of the problem, we have also introduced two new families of inequalities: strenghtened capacity constraints and fork constraints. Computational experiments performed on PDPTW and DARP instances show that both formulations are competitive although the more compact one (in terms of variables) has a slight advantage. In the case of the DARP, comparisons with a previously introduced three-index formulation show that the two new formulations are able to solve much larger instances. The largest instance solved to optimality contains 192 nodes. Given the current state of the art for the exact solution of vehicle routing problems with time windows, it seems fair to say that these are large instances.

Table 7: Impact of valid inequalities for first set of DARP instances

	LP0	LP1	LP2	SEC	SCC	GOC	FC	RC	FULL	U. Bound
a2-16	98.42	99.14	99.93	99.93	99.93	99.93	100.00	100.00	100.00	294.25
a2-20	93.38	95.49	99.31	99.47	99.31	99.31	100.00	100.00	100.00	344.83
a2-24	87.51	95.25	98.81	99.03	98.81	98.81	99.80	99.51	99.80	431.12
a3-24	81.07	88.58	95.62	95.63	95.63	95.62	100.00	99.92	100.00	344.83
a3-30	79.16	88.84	94.55	94.55	94.55	94.52	100.00	100.00	100.00	494.85
a3-36	87.67	92.21	96.85	96.85	96.85	96.85	99.29	98.92	99.29	583.19
a4-32	78.12	85.69	92.23	92.23	92.32	92.23	100.00	100.00	100.00	485.50
a4-40	76.36	92.00	95.64	95.67	95.64	95.64	99.22	99.15	99.32	557.69
a4-48	64.63	86.12	91.96	91.96	91.97	91.96	99.38	98.75	99.62	668.82
a5-40	65.25	84.89	93.10	93.16	93.10	93.10	100.00	99.14	100.00	498.41
a5-50	59.92	78.87	88.40	88.44	88.41	88.40	98.79	97.71	99.04	686.62
a5-60	59.68	75.39	87.18	87.22	87.28	87.18	99.43	98.03	99.48	808.42
a6-48	63.57	79.95	88.25	88.31	88.26	88.26	99.97	98.75	100.00	604.12
a6-60	60.11	75.74	86.22	86.29	86.27	86.22	99.37	98.89	99.61	819.25
a6-72	65.42	79.88	89.08	89.27	89.22	89.09	99.14	98.18	99.36	916.05
a7-56	64.08	81.07	88.12	88.27	88.15	88.12	99.02	98.15	99.21	724.04
a7-70	62.66	77.81	85.74	85.76	85.87	85.74	99.57	98.78	99.75	889.12
a7-84	55.81	71.45	82.27	82.35	82.53	82.28	99.05	97.66	99.20	1033.37
a8-64	66.86	74.63	85.40	85.65	85.40	85.41	99.09	98.13	99.51	747.46
a8-80	58.29	69.87	81.10	81.10	81.19	81.10	99.04	96.17	99.18	945.73
a8-96	53.83	66.81	78.57	78.60	78.67	78.57	97.85	95.03	98.49	1232.61
Avg.	70.56	82.84	90.40	90.46	90.45	90.40	99.43	98.61	99.56	

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council of Canada under grants 227837-04 and OGP0039682. This support is gratefully acknowledged.

References

N. Ascheuer, M. Fischetti and M. Grötschel. "A Polyhedral Study of the Asymmetric Traveling Salesman Problem with Time Windows." *Networks*, **36**:69–79 (2000a).

N. Ascheuer, M. Fischetti and M. Grötschel. "Solving the Asymmetric Travelling Salesman Problem with Time Windows by Branch-and-Cut." *Mathematical Programming*, **90**:475–506 (2001).

N. Ascheuer, M. Jünger and G. Reinelt. "A Branch & Cut Algorithm for the Asymmetric Traveling Salesman Problem with Precedence Constraints." *Computational Optimization and Applications*, **17**:61–84 (2000b).

Table 8: Impact of valid inequalities for second set of DARP instances

	LP0	LP1	LP2	SEC	SCC	GOC	FC	RC	FULL	U. Bound
b2-16	91.76	97.19	99.52	99.53	99.52	99.52	99.59	99.53	99.59	309.41
b2-20	99.90	98.61	100.00	100.00	100.00	100.00	100.00	100.00	100.00	332.64
b2-24	89.30	96.42	99.24	99.24	99.35	99.24	99.96	99.96	99.96	444.71
b3-24	89.65	92.21	95.08	95.08	95.08	95.06	99.33	98.65	99.60	394.51
b3-30	87.91	98.86	99.91	99.91	99.91	99.91	100.00	100.00	100.00	531.44
b3-36	87.72	97.93	99.23	99.23	99.23	99.23	100.00	100.00	100.00	603.79
b4-32	83.82	99.34	100.00	100.00	100.00	100.00	100.00	100.00	100.00	494.82
b4-40	81.25	96.20	98.14	98.14	98.14	98.14	100.00	99.55	100.00	656.63
b4-48	81.75	88.92	95.84	95.82	95.94	95.84	99.67	98.87	99.70	673.81
b5-40	70.43	83.15	91.80	91.67	92.06	91.80	98.99	97.19	99.57	613.72
b5-50	70.78	88.19	92.57	92.44	93.07	92.49	99.20	98.48	99.30	761.40
b5-60	70.75	86.55	91.48	91.52	91.97	91.49	98.88	97.54	99.10	902.04
b6-48	75.90	95.05	98.85	98.82	99.10	98.83	100.00	99.88	100.00	714.83
b6-60	68.75	88.48	94.74	94.75	95.29	94.73	100.00	99.65	100.00	860.07
b6-72	69.99	83.97	90.31	90.36	90.71	90.41	97.76	96.90	98.41	978.47
b7-56	64.54	85.32	90.66	90.67	91.31	90.67	97.56	96.10	98.09	823.97
b7-70	68.25	89.43	94.16	94.24	94.41	94.17	99.44	98.27	99.43	912.62
b7-84	61.59	77.54	86.33	86.47	87.10	86.31	98.78	96.99	99.19	1203.37
b8-64	66.98	84.62	90.83	90.81	91.06	90.86	99.08	98.15	99.44	839.89
b8-80	65.85	88.46	92.18	92.24	92.43	92.20	99.61	98.87	99.63	1036.34
b8-96	59.73	77.36	84.71	84.75	85.30	84.79	97.56	94.36	98.30	1185.55
Avg.	76.50	90.18	94.55	94.56	94.81	94.56	99.30	98.52	99.49	

P. Augerat, J.M. Belenguer, E. Benavent, A. Corberán and D. Naddef. "Separating Capacity Inequalities in the CVRP Using Tabu Search." *European Journal of Operational Research*, **106**:546–557 (1999).

E. Balas, M. Fischetti and W.R. Pulleyblank. "The Precedence-Constrained Asymmetric Traveling Salesman Polytope." *Mathematical Programming*, **68**:241–265 (1995).

J.-F. Cordeau. "A Branch-and-Cut Algorithm for the Dial-a-Ride Problem." *Operations Research* (2006). Forthcoming.

J.-F. Cordeau and G. Laporte. "A Tabu Search Heuristic for the Static Multi-Vehicle Dial-a-Ride Problem." *Transportation Research B*, **37**:579–594 (2003).

J.-F. Cordeau, G. Laporte, J.-Y. Potvin and M.W.P. Savelsbergh. "Transportation on demand." In C. Barnhart and G. Laporte, editors, *Transportation*, Handbooks in Operations Research and Management Science. Elsevier, Amsterdam, 2006. Forthcoming.

T. H. Cormen, C. E. Leiserson and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, Mass., 1990.

Table 9: Results on first set of DARP instances

		(DARP)			(PDPTW1)			(PDPTW2)		
Instance	Cost	Time	Nodes	Cuts	Time	Nodes	Cuts	Time	Nodes	Cuts
a2-16	294.25	0.01	11	39	0.01	0	35	0.01	0	99
a2-20	344.83	0.05	203	91	0.02	0	84	0.02	0	183
a2-24	431.12	0.12	412	131	0.05	3	165	0.04	3	315
a3-24	344.83	1.31	3090	279	0.04	0	233	0.03	0	330
a3-30	494.85	15.48	22250	369	0.08	0	275	0.08	0	564
a3-36	583.19	8.13	8178	415	0.18	5	273	0.18	9	604
a4-32	485.50				0.12	0	482	0.09	0	759
a4-40	557.69				0.51	6	729	0.32	8	896
a4-48	668.82				1.01	12	1120	0.56	4	1848
a5-40	498.41				0.24	0	612	0.17	0	937
a5-50	686.62				2.67	209	1520	1.04	59	1875
a5-60	808.42				2.59	15	1648	1.55	9	2203
a6-48	604.12				0.90	0	1379	0.44	0	1893
a6-60	819.25				3.94	86	1913	1.69	13	2793
a6-72	916.05				8.16	143	2360	3.31	25	3144
a7-56	724.04				5.36	141	2194	1.72	67	2338
a7-70	889.12				6.13	15	2516	3.49	20	3655
a7-84	1033.37				29.43	315	3625	8.23	48	3970
a8-64	747.46				6.48	70	2443	3.61	86	2995
a8-80	945.73				26.14	305	3519	13.22	308	4360
a8-96	1232.61				1210.56	7758	9336	70.55	1652	7436

G. Desaulniers, J. Desrosiers, A. Erdmann, M.M. Solomon and F. Soumis. "VRP with Pickup and Delivery." In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, pages 225–242. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 2002.

M. Desrochers and G. Laporte. "Improvements and Extensions to the Miller-Tucker-Zemlin Subtour Elimination Constraints." *Operations Research Letters*, **10**:27–36 (1991).

J. Desrosiers, Y. Dumas and F. Soumis. "A Dynamic Programming Solution of the Large-Scale Single-Vehicle Dial-a-Ride Problem with Time Windows." *American Journal of Mathematical and Management Sciences*, **6**:301–325 (1986).

Y. Dumas, J. Desrosiers and F. Soumis. "The Pickup and Delivery Problem with Time Windows." *European Journal of Operational Research*, **54**:7–22 (1991).

M. Grötschel and M.W. Padberg. "Polyhedral Theory." In E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, editors, *The Traveling Salesman Problem*, pages 251–305. Wiley, New York, 1985.

Table 10: Results on second set of DARP instances

		(DARP)			(PDPTW1)			(PDPTW2)		
Instance	Cost	Time	Nodes	Cuts	Time	Nodes	Cuts	Time	Nodes	Cuts
b2-16	309.41	0.04	296	109	0.01	3	101	0.01	4	170
b2-20	332.64	0.01	0	30	0.01	0	12	0.01	0	80
b2-24	444.71	0.06	263	107	0.03	2	120	0.03	1	225
b3-24	394.51	0.55	2372	165	0.04	2	147	0.03	3	291
b3-30	531.44	3.20	6267	253	0.05	0	150	0.05	0	313
b3-36	603.79	37.57	44881	246	0.08	0	125	0.08	0	341
b4-32	494.82	75.42	44877	313	0.05	0	136	0.05	0	253
b4-40	656.63				0.15	0	315	0.14	0	609
b4-48	673.81				0.63	9	802	0.50	15	1072
b5-40	613.72				0.59	25	1064	0.28	8	1243
b5-50	761.40				0.92	28	1064	0.66	18	1326
b5-60	902.04				1.96	37	1404	1.81	97	1913
b6-48	714.83				0.27	0	478	0.24	0	613
b6-60	860.07				0.86	0	962	0.68	0	1237
b6-72	978.47				127.71	4769	4864	17.07	989	4278
b7-56	823.97				56.65	1959	7741	13.46	1120	5148
b7-70	912.62				4.66	157	1463	2.12	17	1892
b7-84	1203.37				11.51	77	3036	6.61	58	3463
b8-64	839.89				8.11	405	2155	2.50	66	2165
b8-80	1036.34				4.04	8	1455	3.05	8	1994
b8-96	1185.55				847.41	8246	10171	120.09	2746	8431

Q. Lu and M. Dessouky. "An Exact Algorithm for the Multiple Vehicle Pickup and Delivery Problem." *Transportation Science*, **38**:503–514 (2004).

J. Lysgaard. "Reachability Cuts for the Vehicle Routing Problem with Time Windows." Technical Report L-2004-01, Aarhus School of Business, Denmark, 2004.

D. Naddef and G. Rinaldi. "Branch-and-Cut Algorithms for the Capacitated VRP." In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, pages 53–84. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 2002.

H.N. Psaraftis. "A Dynamic Programming Approach to the Single-Vehicle, Many-to-Many Immediate Request Dial-a-Ride Problem." *Transportation Science*, **14**:130–154 (1980).

H.N. Psaraftis. "An Exact Algorithm for the Single-Vehicle Many-to-Many Dial-a-Ride Problem with Time Windows." *Transportation Science*, **17**:351–357 (1983).

S. Ropke and D. Pisinger. "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows." Technical Report 2004-13, DIKU, University of Copenhagen, 2004.

K.S. Ruland and E.Y. Rodin. "The Pickup and Delivery Problem: Faces and Branch-and-Cut Algorithm." *Computers and Mathematics with Applications*, **33**:1–13 (1997).

M.W.P. Savelsbergh and M. Sol. "DRIVE: Dynamic Routing of Independant Vehicles." *Operations Research*, **46**:474–490 (1998).

Chapter 9

Branch-and-Cut-and-Price for the Pickup and Delivery Problem with Time Windows

Branch-and-Cut-and-Price for the Pickup and Delivery Problem with Time Windows

Stefan Ropke
DIKU, University of Copenhagen, Denmark
sropke@diku.dk

Jean-François Cordeau
Canada Research Chair in Distribution Management, HEC Montréal
3000, chemin de la Côte-Sainte-Catherine, Montréal, H3T 2A7 Canada
jean-francois.cordeau@hec.ca

Abstract

In the pickup and delivery problem with time windows (PDPTW), vehicle routes must be designed to satisfy a set of transportation requests, each involving a pickup and a delivery location, under capacity, time window and precedence constraints. This paper introduces a branch-and-cut-and-price algorithm in which lower bounds are computed by solving the linear programming relaxation of a set-partitioning formulation. This relaxation is solved by a column generation scheme with an elementary shortest path pricing problem. Various relaxations, yielding different pricing problems, are also investigated, and valid inequalities are proposed to strengthen some of these relaxations. The strength of the different relaxations is investigated through extensive computational experiments. These experiments also show that the proposed method outperforms a recent branch-and-cut algorithm. The largest problem instance solved contains 500 requests.
Keywords: pickup and delivery; time windows; branch-and-cut-and-price.

1 Introduction

In the classical Vehicle Routing Problem (VRP), a fleet of vehicles based at a common depot must be routed to visit exactly once a set of customers with known demand. Each vehicle route must start and finish at the depot and the total demand of the customers visited by the route must not exceed the vehicle capacity. In the VRP with Time Windows (VRPTW), a time window is associated with each customer and the vehicle visiting a given customer cannot arrive after the end of the time window. The Pickup and Delivery Problem with Time Windows (PDPTW) is a further generalization of the VRP in which each customer request is associated with two locations: an origin location where a certain demand must be picked up and a destination where this demand must be delivered. Each route must also satisfy pairing and precedence constraints: for each request, the origin must precede the destination, and both locations must be visited by the same vehicle. The VRPTW can be seen as a special case of the PDPTW in which all requests have a common origin which corresponds to the depot.

The PDPTW has applications in various contexts such as urban courier services, less-than-truckload transportation, and door-to-door transportation services for the elderly and the disabled. In the latter case, narrow time windows are often considered and ride time constraints are imposed to control the time spent by a passenger in the vehicle. The resulting problem is called the Dial-a-Ride Problem (DARP) and will also be addressed in this paper.

The VRP and VRPTW are well known combinatorial optimization problems which have received a lot of attention (see, e.g., Toth and Vigo, 2002). Since it generalizes the VRPTW, the PDPTW is clearly $\mathcal{NP}$-hard. Over the last few decades, several heuristics have been proposed for both the PDPTW and the closely-related DARP. However, because of the difficulty of these problems, work on exact methods has been somewhat limited.

Two main approaches have been used to solve the PDPTW exactly: branch-and-price and branch-and-cut. Branch-and-price methods (see, e.g., Barnhart et al., 1998; Desaulniers et al., 1998) use a branch-and-bound scheme in which lower bounds are computed by column generation. The first branch-and-price algorithm for the PDPTW was proposed by Dumas et al. (1991) who considered a set-partitioning formulation of the problem in which each each column corresponds to a feasible vehicle route and each constraint is associated to a request that must be satisfied exactly once. The resulting pricing subproblem is a shortest path problem with time window, capacity, pairing and precedence constraints. This problem is solvable by dynamic programming and the authors use an algorithm similar to the one developed by Desrosiers et al. (1986) for the single-vehicle pickup and delivery problem with time windows. Several label elimination methods are proposed to accelerate the dynamic programming algorithm, and arc elimination rules are used to reduce the size of the problem. The authors point out that their approach works well when the demand of each customer is large with respect to vehicle capacity. The largest instance solved with their approach contains 55 requests.

Another branch-and-price approach for the PDPTW was later described by Savelsbergh and Sol (1998). Their approach differs from that of Desrosiers et al. in several respects: *i)* whenever possible, they use construction and improvement heuristics to solve the pricing subproblem; *ii)* a sophisticated column management mechanism is used to keep the column generation master problem as small as possible; *iii)* columns are selected with a bias toward increasing the likelihood of identifying feasible integer solutions during the solution of the master problem; *iv)* branching decisions are made on additional variables representing the fraction of a request that is served by a given vehicle; and *v)* a primal heuristic is used at each node of the search tree to compute upper bounds.

Column generation was also used recently by Xu et al. (2003) and Sigurd et al. (2004) to address variants of the PDPTW arising in long-haul transportation planning and in the transportation of live animals, respectively.

The second family of exact approaches for the PDPTW is branch-and-cut. In branch-and-cut, valid inequalities (i.e., cuts) are added to the formulation at each node of the branch-and-bound tree to strengthen the relaxations which are usually solved by the simplex algorithm. Relying on the previous work of Balas et al. (1995) and Ruland and Rodin (1997) on the Precedence-Constrained Traveling Salesman Problem (PCTSP) and the TSP with Pickup and Delivery (TSPPD), Cordeau (2005) developed a branch-and-cut algorithm for the DARP based on a three-index formulation of the problem. This algorithm was able to solve instances with four vehicles and 32 requests. It was later improved by Ropke et al. (2005) who compared different formulations of the DARP and PDPTW, and introduced two new families of inequalities for these problems. One is an adaptation of the *reachability cuts* introduced by Lysgaard (2005) for the VRPTW, while the other is called *fork inequalities*. Both families can also be used in the context of column generation and will be described in Section 4. Using these inequalities, Ropke et al. were able to solve instances with eight vehicles and 96 requests. Another branch-and-cut approach, based on a two-index formulation was proposed by Lu and Dessouky (2004). This formulation contains a polynomial number of constraints, but relies on extra variables to impose pairing and precedence constraints. Instances with up to five vehicles and 25 requests were solved optimally with this approach.

For reviews on pickup and delivery problems, the reader is referred to the works of Savelsbergh and Sol (1995), Desaulniers et al. (2002) and Cordeau et al. (2005).

In this paper, we introduce a new branch-and-cut-and-price algorithm for the PDPTW and the DARP. It is well known that set partitioning formulations of vehicle routing problems tend to provide stronger lower bounds than formulations based on arc (flow) variables (see Bramel and Simchi-Levi, 2002). Unfortunately, the column generation pricing subproblem used with this formulation is often an elementary resource-constrained shortest path problem which is $\mathcal{NP}$–hard. Our aim is to investigate various relaxations of the pricing subproblem. These relaxations are still NP-hard but in practice they can be easier to solve to optimality.

The contributions of the paper are threefold. First, we analyze the different relaxations in terms of lower bound quality and computational difficulty. Second, we show that valid inequalities can be introduced in the formulation to improve the quality of some of the lower bounds. Third, we report extensive computational experiments on several sets of test instances from the literature and introduce new large-scale instances.

The remainder of the paper is organized as follows. Section 2 defines the PDPTW and introduces mathematical formulations of the problem. Section 3 discusses possible pricing subproblems that can be used within a branch-and-price algorithm, while Section 4 describes valid inequalities that can be added to the formulation. The resulting branch-and-cut-and-price algorithm is then described in Section 5. Finally, computational results are reported in Section 6, followed by conclusions in the last section.

2 Mathematical Formulation

In this section, we introduce the notation that is used throughout the paper. We then present a classical three-index model of the problem, followed by a set-partitioning formulation.

2.1 Notation

Let n denote the number of requests to satisfy. We define the PDPTW on a directed graph $G = (N, A)$ with node set $N = \{0, \ldots, 2n+1\}$ and arc set A. Nodes 0 and $2n+1$ represent the origin and destination depots while subsets $P = \{1, \ldots, n\}$ and $D = \{n+1, \ldots, 2n\}$ represent pickup and delivery nodes, respectively. With each request i are thus associated a pickup node i and a delivery node $n+i$.

With each node $i \in N$ are associated a load q_i and a non-negative service duration d_i satisfying $q_0 = q_{2n+1} = 0$, $q_i = -q_{n+i}$ $(i = 1, \ldots, n)$ and $d_0 = d_{2n+1} = 0$. A time window $[a_i, b_i]$ is also associated with every node $i \in P \cup D$, where a_i and b_i represent the earliest and latest time, respectively, at which service may start at node i. The depot nodes may also have time windows $[a_0, b_0]$ and $[a_{2n+1}, b_{2n+1}]$ representing the earliest and latest times, respectively, at which the vehicles may leave from and return to the depot. Let K denote the set of vehicles. We assume that vehicles are identical and have capacity Q. With each arc $(i, j) \in A$ are associated a routing cost c_{ij} and a travel time t_{ij}. In the remainder of the paper, we assume that the travel time t_{ij} includes the service time d_i at node i. We also assume that the triangle inequality holds both for routing costs and travel times.

2.2 Three-index formulation of the PDPTW

For each arc $(i, j) \in A$ and each vehicle $k \in K$, let x_{ij}^k be a binary variable equal to 1 if and only if vehicle k travels directly from node i to node j. For each node $i \in N$ and each vehicle $k \in K$, let B_i^k be the time at which vehicle k begins service at node i, and Q_i^k be the load of vehicle k immediately after visiting node i. Using these variables, the PDPTW can be formulated as the following mixed-integer program:

$$\text{Min} \quad \sum_{k \in K} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij}^k \tag{1}$$

subject to

$$\sum_{k \in K} \sum_{j \in N} x_{ij}^k = 1 \qquad \forall i \in P \tag{2}$$

$$\sum_{j \in N} x_{ij}^k - \sum_{j \in N} x_{n+i,j}^k = 0 \qquad \forall i \in P, k \in K \tag{3}$$

$$\sum_{j \in N} x_{0j}^k = 1 \qquad \forall k \in K \tag{4}$$

$$\sum_{j \in N} x_{ji}^k - \sum_{j \in N} x_{ij}^k = 0 \qquad \forall i \in P \cup D, k \in K \tag{5}$$

$$\sum_{i \in N} x_{i,2n+1}^k = 1 \qquad \forall k \in K \tag{6}$$

$$B_j^k \geq (B_i^k + t_{ij}) x_{ij}^k \qquad \forall i \in N, j \in N, k \in K \tag{7}$$

$$Q_j^k \geq (Q_i^k + q_j) x_{ij}^k \qquad \forall i \in N, j \in N, k \in K \tag{8}$$

$$B_i^k + t_{i,n+i} \leq B_{n+i}^k \qquad \forall i \in P \tag{9}$$

$$a_i \leq B_i^k \leq b_i \qquad \forall i \in N, k \in K \tag{10}$$

$$\max\{0, q_i\} \leq Q_i^k \leq \min\{Q, Q + q_i\} \qquad \forall i \in N, k \in K \tag{11}$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall i \in N, j \in N, k \in K. \tag{12}$$

The objective function (1) minimizes the total routing cost. Constraints (2) and (3) ensure that each request is served exactly once and that the pickup and delivery nodes are visited by the same vehicle. Constraints (4)-(6) guarantee that the route of each vehicle k starts at the origin depot and ends at the destination depot. Consistency of the time and load variables is ensured by constraints (7) and (8). Constraints (9) ensure that for each request i, the pickup node is visited before the delivery node. Finally, inequalities (10) and (11) impose time windows and capacity constraints, respectively.

2.3 Set partitioning formulation of the PDPTW

To formulate the problem as a set partitioning problem, let Ω denote the set of all feasible routes satisfying constraints (3)-(12), dropping index k (as all vehicles are identical). Alternatively the routes should satisfy (3)-(12) for a particular $k \in K$. For each route $r \in \Omega$, let $\tilde{c}_r$ be the cost of the route and let a_{ir} be a constant indicating the number of times node $i \in P$ is visited by r. Let also y_r be a binary variable equal to 1 if and only if route $r \in \Omega$ is used in the solution. The PDPTW can then be formulated as the following set partitioning problem:

$$\text{Min} \quad \sum_{r \in \Omega} \tilde{c}_r y_r \tag{13}$$

subject to

$$\sum_{r \in \Omega} a_{ir} y_r = 1 \qquad \forall i \in P \tag{14}$$

$$y_r \in \{0,1\} \qquad \forall r \in \Omega. \tag{15}$$

The objective function (13) minimizes the cost of the chosen routes while constraints (14) ensure that every request is served once. A lower bound on the optimal value of (13)-(15) can be obtained by solving the linear programming (LP) relaxation which is obtained by replacing (15) with the simple bound constraints $0 \leq y_r \leq 1 \; \forall r \in \Omega$.

Because of the large size of set Ω, it is usually very difficult to solve or even to represent model (13)-(15) explicitly. Instead its LP relaxation is solved using column generation. In a column generation approach, a restricted master problem is obtained by considering a subset $\bar{\Omega} \subseteq \Omega$ of routes. Additional columns of negative reduced-cost are generated by solving a *pricing subproblem*. Following Wolsey (1998), we call the problem defined by (13)–(15) the *integer programming master problem* (*IPM*) and its LP relaxation the *linear programming master problem* (*LPM*). The pricing problem for the PDPTW is

$$\text{Min} \quad \sum d_{ij} x_{ij} \tag{16}$$

subject to constraints (3)–(12) (dropping index k), where d_{ij} is defined as

$$d_{ij} = \begin{cases} c_{ij} - \pi_i & \forall i \in P, j \in N \\ c_{ij} & \forall i \in N \setminus P, j \in N, \end{cases} \tag{17}$$

and π_i is the dual variable associated with the set partitioning constraint (14) for node i.

The definition of d_{ij} in equation (17) ensures that $d_{ij} + d_{jk} \geq d_{ik}$ if j is a delivery node. As will be shown in Section 3 this is computationally convenient. We denote this problem as SP1. The problem defined by objective (16) and constraints (3)–(12) is a constrained shortest path problem called the *Elementary Shortest Path Problem with Time Windows, Capacity, and Pickup and Delivery* (ESPPTWCPD). In Section 3 we explain how this and related problems can be solved using label setting algorithms.

Instead of solving the shortest path problem SP1 one can solve relaxed versions of this problem. A relaxed shortest path problem implies that a set of routes Ω' is implicitly considered, where $\Omega \subseteq \Omega'$. If Ω' satisfies the property that none of the columns from the set

$\Omega' \setminus \Omega$ can be used in a feasible integer solution to IPM then the set partitioning problem solved on the set Ω' will have the same set of optimal solutions as the one solved on Ω. Obviously, the lower bound obtained by solving the LP relaxation on Ω' may, however, be weaker. An example of a relaxation of the shortest path problem that satisfies this property consists of allowing cycles in the path. In this case, some requests may be served more than once. Paths containing cycles cannot, however, appear in a feasible integer solution because of constraints (14). This relaxation was used by Dumas et al. (1991) and it is described in more detail in Section 3.3.

Relaxations inducing sets Ω' for which one cannot ensure that no column from $\Omega' \setminus \Omega$ can belong to a feasible integer solution to IPM can also be used. In this case, however, valid inequalities must be added to the master problem to render such solutions infeasible.

Valid inequalities expressed in terms of the x_{ij}^k variables from the three-index formulation (1)-(12) can be added to the master problem following the approach proposed by Kohl et al. (1999) for the VRPTW. We first observe that such inequalities can be expressed in terms of x_{ij} variables as all vehicles are identical ($x_{ij} = \sum_{k \in K} x_{ij}^k$). The inequalities can be written in the form

$$\sum_{i=0}^{2n+1} \sum_{j=0}^{2n+1} \alpha_{ij} x_{ij} \geq \beta,$$

where $\alpha_{ij} \in \mathbb{R}$ is the coefficient of arc $(i,j) \in A$ and $\beta \in \mathbb{R}$ is a constant. This inequality is transfered to the master problem as

$$\sum_{r \in \Omega} \phi_r y_r \geq \beta,$$

where $\phi_r = \sum_{(i,j) \in r} \alpha_{ij}$. The notation $(i,j) \in r$ means that arc $(i,j) \in A$ is used in route r. It is easy to see that the introduction of a valid inequality in the master problem modifies the pricing problem. Indeed, the arc costs d_{ij} are now defined as follows:

$$d_{ij} = \begin{cases} c_{ij} - \pi_i - \alpha_{ij}\mu & \forall i \in P, j \in N \\ c_{ij} - \alpha_{ij}\mu & \forall i \in N \setminus P, j \in N, \end{cases} \tag{18}$$

where μ is the dual variable associated with the added inequality. Any number of inequalities can be added in this way. Notice that this definition of d_{ij} does not guarantee $d_{ij} + d_{jk} \geq d_{ik}$ when j is a delivery node, as it was the case with definition (17).

3 Constrained Shortest Path Problems

Resource constrained shortest path problems arising in column generation approaches for vehicle routing problems are typically solved using dynamic programming techniques called labeling algorithms. Notice that the term "shortest path" should be interpreted carefully: given a cost function which can itself be viewed as a resource, one wishes to find the least-cost feasible path from the source node to the sink node. An overview of constrained shortest path problems and of appropriate labeling algorithms for their solution is given by Irnich and Desaulniers (2005).

In this section we will show how the ESPPTWCPD introduced in section 2.3 can be solved using a labeling algorithm. Three relaxations of the problem is considered in sub-section 3.3 – 3.5.

3.1 Label setting shortest path algorithms

Consider a weighted directed graph $G = (V, A)$ where V is the set of nodes, A is the set of arcs, s is the source node and t is the sink node. We assume that no arc enters node s and no arc leave node t. Let γ be the number of resources in the problem. Traversing arcs "consumes" resources. Let $f_{ij}^p \in \mathbb{Q}$ denote the consumption of resource $p \in \{1, \ldots, \gamma\}$ for arc $(i, j) \in A$. For every node $i \in V$ lower bounds $l_i^p \in \mathbb{Q}$ and upper bounds $u_i^p \in \mathbb{Q}$ on the resource variables $p \in \{1, \ldots, \gamma\}$ are given.

In label setting shortest path algorithms, a label consists of three elements: a node, the cumulated resource consumption at that node, and a pointer to its parent label. A label $L = (i, R, p)$ corresponds to a path starting at node s and ending at node i with a certain resource consumption characterized by the vector $R \in \mathbb{Q}^\gamma$. The parent label p is necessary to reconstruct the path between s and i. Resource constrained shortest path problems can be solved using an algorithm based on the pseudo-code presented in Algorithm 1.

Algorithm 1 Pseudo code for labeling algorithm

1 `Input: graph` $G = (V, A)$`, source node` s`, sink node` t
2 $U = \{(s, (l_s^1, \ldots, l_s^\gamma), nil)\}$
3 `while` $U \neq \emptyset$ `do`
4   L `= removefirst(`U`)`
5   i = node(L)
6   `if no label in` $\mathcal{L}_i$ `dominates` L `then`
7     $\mathcal{L}_i = \mathcal{L}_i \cup \{L\}$
8   `extend` L `along all arcs` (i, j) `leaving node` i
9   `add all feasible extensions to` U
10 `return path corresponding to best label in` $\mathcal{L}_t$

In line 2 of the algorithm, an initial label $(s, (l_s^1, \ldots, l_s^\gamma), nil)$ corresponding to the source node s is created. In this label, the resource consumption is set according to the lower bounds for node s. Here, U designates the set of unprocessed labels and $\mathcal{L}_i$ is the set of processed labels at node i (paths ending at node i). Lines 4 to 8 are repeated as long as there are unprocessed labels. In line 4 a new unprocessed label is selected using the `removefirst` function (the function removes the label from U). In line 5 the node of the label is retrieved and line 6 checks whether the label can be discarded (this is explained in more detail below). If the label cannot be discarded then it it is stored in the set of processed labels for node i in line 7. In line 8, new labels are created by extending label L. Extending a label $L = (i, R, p)$ along arc (i, j) results in the label (j, R', L) where the kth component R'_k of R' is given by either $R'_k = \max\left(l_j^k, R_k + f_{ij}^k\right)$ or $R'_k = R_k + f_{ij}^k$ depending on the type of resource. The new label is feasible if all resource variables are within their lower and upper bounds for node j. All labels corresponding to feasible extensions of label L is added to U in line 9. In line 10, the label with the least cost at the sink node is returned.

To guarantee the termination of the algorithm it is sufficient to assume the following: *i*) it should be possible to define an ordering of all labels such that given two distinct labels, one should be strictly *greater* than the other according to the chosen ordering; *ii*) the process of extending a label (line 7) should result in a *greater* label according to the ordering; and *iii*) there should exist a maximal label such that all other labels are *smaller* than or equal to that label. If the function `removefirst` returns the *smallest* label in U according to the ordering then the algorithm will terminate. First, we will never process a label twice as we choose the smallest unprocessed label and extending it yields greater labels. Second, because we have an upper bound on the possible labels and because each extension results in a greater label, the process will terminate as it will eventually reach the upper bound.

Without the test in line 6 the algorithm is a brute-force approach that enumerates all feasible paths. The test in line 6 removes unpromising labels based on a so called *dominance criterion*. We say that label L_1 *dominates* label L_2, written $L_1 \preceq_{dom} L_2$ if and only if they are assigned to the same node and no feasible extension of the path corresponding to L_2 with a path to t has a lower cost than the best (with respect to cost) feasible extension of the path corresponding to L_1 with a path to t. If $L_1 \preceq_{dom} L_2$ then there is no need to consider L_2, and we need only examine extensions of L_1.

Given two labels it can be difficult to determine whether one label dominates the other as we potentially have to examine all possible augumentations of the corresponding paths to node t. Consequently we use simpler criteria which for some pairs of labels (L_1, L_2) cannot determine whether one of the labels dominates the other. In section 3.2 and 3.3 we describe examples of such criteria.

3.2 ESPPTWCPD - SP1

The ESPPTWCPD, denoted SP1, is the natural pricing problem for the PDPTW and the one that provides the best lower bounds. In the context of the PDPTW, it was first used by Sol (1994) and later by Sigurd et al. (2004) for a PDPTW with additional precedence constraints. Sigurd et al. (2004) described a general labeling algorithm for the ESPPTWCPD and a more efficient one that takes advantage of the additional precedence constraints.

In this section we present a new labeling algorithm for the ESPPTWCPD which contains a better dominance criteria compared to the algorithm proposed by Sol (1994) and the general one described by Sigurd et al. (2004).

In what follows we assume that the source and sink nodes are, respectively, 0 and $2n+1$, as is the case in the shortest path problems that must be solved as pricing problems in our column generation algorithm.

3.2.1 Label management

Table 1 summarizes the data that are stored for each label (the parent label is omitted in this table). Thus, t, l, c, $\mathcal{V}$ and $\mathcal{O}$ are resources. The notation $t(L)$ is used to refer to the arrival time in label L and similar notation is used for the rest of the resources. The notation $\mathcal{P}(L)$ represents the path corresponding to L and (p_1, p_2) represents the path obtained by concatenating path p_2 on path p_1.

When extending a label L along an arc $(\eta(L), j)$, the extension is legal only if

$$t(L) + t_{\eta(L),j} \leq b_j \quad (19)$$

$$l(L) + q_j \leq Q. \quad (20)$$

Inequality (19) ensures time window feasibility while inequality (20) ensures capacity

Table 1: Resources used in SP1

η	The node of the label
t	The arrival time at the node
l	The current load
c	The current cost
$\mathcal{V}$	$\mathcal{V} \subseteq \{1, \ldots, n\}$ is the set of requests that have been started on the path (and possibly finished).
$\mathcal{O}$	$\mathcal{O} \subseteq \{1, \ldots, n\}$ is the set of requests that have been started but not finished, i.e., the pickup has been served but not the delivery. The requests in $\mathcal{O}$ are said to be *open*.

feasibility. Furthermore, L and j must satisfy one of the following three conditions:

$$0 < j \leq n \quad \wedge \quad \mathcal{V}(L) \cap \{j\} = \emptyset \tag{21}$$

$$n < j \leq 2n \quad \wedge \quad j \in \mathcal{O}(L) \tag{22}$$

$$j = 2n + 1 \quad \wedge \quad \mathcal{O}(L) = \emptyset. \tag{23}$$

Condition (21) ensures that if j is a pickup node then that node must not have been visited before on the path. This is to ensure that the the algorithm finds an elementary path. Condition (22) ensures that if j is a delivery node then the path must have already visited the corresponding pickup node, i.e., the precedence relationship between pickups and deliveries is satisfied. Finally, condition (23) ensures that if j is the sink node then all requests that have been started have also been finished. This condition enforces the pairing constraint: the pickup and delivery from any given request must be served on the same path. In the presence of (22), condition (21) is sufficient to ensure that only elementary paths are considered.

If extension along the arc $(\eta(L), j)$ is feasible then a new label L' is created at node j. The information in label L' is set as follows:

$$\eta(L') = j \tag{24}$$

$$t(L') = max\{a_j, t(L) + t_{\eta(L),j}\} \tag{25}$$

$$l(L') = l(L) + q_j \tag{26}$$

$$c(L') = c(L) + d_{\eta(L),j} \tag{27}$$

$$\mathcal{V}(L') = \begin{cases} \mathcal{V}(L) \cup \{j\} & \text{if } j \in P \\ \mathcal{V}(L) & \text{if } j \in D \end{cases} \tag{28}$$

$$\mathcal{O}(L') = \begin{cases} \mathcal{O}(L) \cup \{j\} & \text{if } j \in P \\ \mathcal{O}(L) \setminus \{j - n\} & \text{if } j \in D. \end{cases} \tag{29}$$

Equations (24)-(27) set the current node, the time, the load and the cost of the new label, respectively. Equation (28) updates the set of visited requests. Node j is only added if it is a pickup node. Equation (29) updates the set of open requests. If a pickup (resp. delivery) node is visited, the corresponding request is added to (resp. removed from) the set to indicate that the request has been started (resp. completed).

3.2.2 Dominance criterion

The dominance criterion employed in this section is the following: a label L_1 dominates a label L_2 if

$$\eta(L_1) = \eta(L_2), \quad t(L_1) \leq t(L_2), \quad c(L_1) \leq c(L_2), \quad \mathcal{V}(L_1) \subseteq \mathcal{V}(L_2), \quad \mathcal{O}(L_1) \subseteq \mathcal{O}(L_2).$$

We denote this criterion (DOM1).

Proposition 1. DOM1 is a valid dominance criterion when considering the definition of d_{ij} from equation (17).

Proof. The proof follows from that of Proposition 4 in Dumas et al. (1991). Let p be an optimal (with respect to cost) and feasible path extending the path of L_2 to $2n+1$. If such a path does not exist then clearly one can remove label L_2. Let p' be the path obtained from p by removing the deliveries corresponding to the requests in $\mathcal{O}(L_2) \setminus \mathcal{O}(L_1)$. As $(\mathcal{P}(L_1), p)$ is feasible, then so is $(\mathcal{P}(L_2), p')$. Indeed, it is easy to see that it is feasible with respect to time windows because travel times satisfy the triangle inequality. The capacity is not violated on $(\mathcal{P}(L_2), p')$ as it was not violated on $(\mathcal{P}(L_1), p)$ and $\mathcal{P}(L_2)$ does not visit the pickups corresponding to the deliveries removed from p. It is also easy to see that $(\mathcal{P}(L_2), p')$ is elementary and satisfies pairing constraints. The cost of $(\mathcal{P}(L_2), p')$ is less than or equal to the cost of $(\mathcal{P}(L_1), p)$ because $c(L_1) \leq c(L_2)$ and the cost of p' is less than or equal to the cost of p because removing deliveries cannot increase the cost of a path due to the definition of d_{ij} in equation (17) and the triangle inequality on c_{ij}. As a result, the best (with respect to cost) extension of label L_1 to $2n+1$ will always be better than the best extension of L_2 to $2n+1$. Hence, label L_1 dominates label L_2.$\square$

Notice that it is not necessary to consider the load of a label in the dominance criterion. Indeed, since $\mathcal{O}(L_1) \subseteq \mathcal{O}(L_2)$ then the load of label L_1 must be smaller than that of L_2.

In the labeling algorithm of Sol (1994), labels contain the cost c, the arrival time t, and the sets $S^+ \subseteq \{1, \ldots, n\}$ and $S^- \subseteq \{1, \ldots, n\}$. Here, S^+ is the set of requests that have been picked up and S^- is the set of requests that have been delivered. With respect to the sets $\mathcal{V}$ and $\mathcal{O}$, one obtains $S^+ = \mathcal{V}$ and $S^- = \mathcal{V} \setminus \mathcal{O}$. In terms of S^+ and S^- the dominance criterion proposed in this paper is the following: label L_1 dominates label L_2 if

$$\eta(L_1) = \eta(L_2), \quad t(L_1) \leq t(L_2), \quad c(L_1) \leq c(L_2),$$

$$S^+(L_1) \subseteq S^+(L_2), \quad \left(S^+(L_1) \setminus S^-(L_1)\right) \subseteq \left(S^+(L_2) \setminus S^-(L_2)\right)$$

Sol (1994) used the following criterion:

$$\eta(L_1) = \eta(L_2), \quad t(L_1) \leq t(L_2), \quad c(L_1) \leq c(L_2),$$

$$S^+(L_1) = S^+(L_2), \quad S^-(L_1) = S^-(L_2).$$

If label L_1 dominates label L_2 according to the criterion used by Sol (1994) then it also dominates L_2 using the criterion proposed in this paper, but the converse is not true. Our new criterion is therefore stronger than the one used by Sol.

Given a label L, let $U(L)$ be the set of *unreachable* requests from $\mathcal{P}(L)$. This set is defined as follows: $U(L) = \mathcal{V}(L) \cup \left\{i \in \{1, \ldots, n\} : t(L) + t_{\eta(L),i} > b_i\right\}$.

By replacing $\mathcal{V}(L)$ with $U(L)$ in (DOM1) and in equation (21) and (28), one obtains a stronger dominance criterion (DOM1'). This new dominance criterion is stronger for the following reason: if a label L_1 dominates a label L_2 according to (DOM1) it also dominates L_2 according to the new criterion (DOM1'), but the converse is not true. In order to prove the validity of the new dominance criterion one has to consider the case where a label L_1 dominates a label L_2 according to the new criterion, but not according to (DOM1). That is when $U(L_1) \subseteq U(L_2)$ but $\mathcal{V}(L_1) \nsubseteq \mathcal{V}(L_2)$. Define $W = \mathcal{V}(L_1) \setminus \mathcal{V}(L_2)$. Any extension of $\mathcal{P}(L_1)$ cannot visit the requests in W. Hence, if an extension of $\mathcal{P}(L_2)$ could visit one request $i \in W$ then there could be an extension of $\mathcal{P}(L_2)$ that would be better than any extension of $\mathcal{P}(L_1)$ if π_i is large. To see that no extension of $\mathcal{P}(L_2)$ can visit requests in W observe that $W \subseteq U(L_2)$ since $W \subseteq \mathcal{V}(L_1) \subseteq U(L_1) \subseteq U(L_2)$. As a result, any extension of $\mathcal{P}(L_2)$ that visits a node from W is violating a time window because of the definition of $U(L)$ and the assumption that t_{ij} satisfies the triangle inequality.

The idea of considering $U(L)$ instead of $\mathcal{V}(L)$ was proposed by Feillet et al. (2004) in the context of the pricing problem for the VRPTW.

The dominance criteria (DOM1) and (DOM1') are strong, but they give rise to strict requirements on the cost structure of the underlying network. The definition of d_{ij} from equation (18) cannot be used together with (DOM1) and (DOM1'). Indeed, one cannot ensure that the removal of a delivery node from a sub path will reduce the cost, and this property is used in the proof of Proposition 1. Consequently, the dominance criterion cannot be used directly in the presence of additional valid inequalities. Another drawback of these dominance criteria is that the removing of arcs from the network must be performed very carefully. An arc (i, j) cannot be removed if the sub path i, k, j is valid for some delivery k. In this case one cannot argue that removing deliveries from a path will yield a path with lower cost since removing the deliveries will result in an invalid route. Arc elimination is often useful within a branch-and-bound scheme that branches on the arcs in the original formulation (1)-(12).

As a consequence of the above discussion, we define an alternative dominance criterion (DOM1†) as follows: a label L_1 dominates a label L_2 if

$$\eta(L_1) = \eta(L_2), \quad t(L_1) \le t(L_2), \quad c(L_1) \le c(L_2), \quad U(L_1) \subseteq U(L_2), \quad \mathcal{O}(L_1) = \mathcal{O}(L_2).$$

This criterion is not as strong as (DOM1'), but it is easier to use, as it does not rely on any specific assumption regarding the cost structure of the network.

3.2.3 Label elimination

Dumas et al. (1991) proposed rules for eliminating labels that cannot be extended to node $2n + 1$. The key observation is that given a label L one can examine the deliveries of the open requests in $\mathcal{O}(L)$. If it is impossible to create a path from $\eta(L)$ through the deliveries of $\mathcal{O}(L)$ to node $2n+1$ that satisfies all time windows, then label L can be discarded because of the triangle inequality on t_{ij}. Determining whether such a path exists can be done by solving a traveling salesman problem with time windows which is $\mathcal{NP}$-hard. Consequently, Dumas et al. (1991) proposed to consider only subsets of $\mathcal{O}(L)$ of cardinality one and two. We are going to use the same approach. Furthermore we also test a single subset containing three deliveries. The first delivery i_1 in this subset is the one farthest from $\eta(L)$, the next delivery i_2 is the one farthest from $\eta(L)$ and i_1 and the last delivery is the one farthest from $\eta(L)$, i_1 and i_2.

3.3 SPPTWCPD - SP2

We now consider the *Shortest Path Problem with Time Windows, Capacity, and Pickup and Delivery* (SPPTWCPD), denoted SP2, which relaxes SP1 by not requiring paths to be elementary. In this problem we do, however, impose two conditions which help prevent cycles: *i)* after performing a pickup, the same pickup cannot be performed again before the corresponding delivery has been performed, and *ii)* a delivery cannot be performed before the corresponding pickup has been performed. These conditions ensure that any cycle in a path will contain at least four nodes. The shortest cycle is of the form $i \to n+i \to j \to i$. One cannot go from $n+i$ to i as the corresponding arc does not exist in our graph (see Section 5.2 for details on preprocessing). If time windows are tight, such cycles are unlikely to arise and the SPPTWCPD should yield good lower bounds. This shortest path problem was used as a pricing problem by Dumas et al. (1991).

3.3.1 Label management

For SP2, we store for each label the data summarized in Table 2.

Table 2: Resources used in SP2

η	The node of the label
t	The arrival time at the node
l	The current load
c	The current cost
$\mathcal{O}$	$\mathcal{O} \subseteq \{1, \ldots, n\}$ is the set of requests that have been started but not finished.

Determining if an extension of a label is feasible and creating new labels is done in a similar way as for SP1. We do not, however, maintain the set $\mathcal{V}$. Hence, equation (28) is not used and equation (21) is replaced with

$$0 < j \leq n \quad \wedge \quad \mathcal{O}(L) \cap \{j\} = \emptyset. \tag{30}$$

Replacing equation (21) with (30) implies that non elementary paths can be generated. When the delivery of request i has been performed, i is removed from $\mathcal{O}$ according to equation (29) and the path may then visit the pickup node of request i once again.

3.3.2 Dominance criterion

The dominance criterion employed, denoted (DOM2) is the following: a label L_1 dominates a label L_2 if

$$\eta(L_1) = \eta(L_2), \quad t(L_1) \leq t(L_2), \quad c(L_1) \leq c(L_2), \quad \mathcal{O}(L_1) \subseteq \mathcal{O}(L_2).$$

Dumas et al. (1991) showed that this dominance criterion is valid.

Criterion (DOM2) has the same weaknesses as (DOM1) and (DOM1'), i.e., it can only be used on a network that satisfies certain assumptions on the cost structure, and therefore cannot be used directly in a branch-and-cut-and-price algorithm. As a result, we again resort to a weaker criterion that allows arbitrary cost structures:

$$\eta(L_1) = \eta(L_2), \quad t(L_1) \leq t(L_2), \quad c(L_1) \leq c(L_2), \quad \mathcal{O}(L_1) = \mathcal{O}(L_2).$$

We denote this dominance criterion (DOM2†).

The label elimination rules described in Section 3.2.3 can be used for the SPPTWCPD as well. It is actually in this context that they were first introduced by Dumas et al.

3.4 ESPPTW - SP3

The *Elementary Shortest Path Problem with Time Windows* (ESPPTW), denoted as SP3, relaxes SP1 by removing the capacity, precedence and pairing constraints. As a result, the shortest path may visit the pickup node of a given request without visiting the corresponding delivery node, and vice-versa. In addition, if both the pickup and delivery nodes of a request are visited then the pickup node may be visited after the delivery node.

This problem is also a relaxation of the Elementary Shortest Path Problem with Time Windows and Capacity (ESPPTWC) which was recently used with success as a pricing problem for the VRPTW (Chabrier, 2003; Feillet et al., 2004).

If SP3 is used as a pricing subproblem for the PDPTW, the set partitioning formulation (13)-(15) must be modified to include one constraint for each node in $P \cup D$ (instead of only one constraint for each pickup node).

3.4.1 Label management

We store for each label the data summarized in Table 3.

Table 3: Resources used in SP3

η	The node of the label
t	The arrival time at the node
c	The current cost
$\mathcal{V}$	$\mathcal{V} \subseteq \{1, \ldots, 2n\}$ is the nodes that have been visited on the path.

A label L can be extended to a node j if

$$t(L) + t_{\eta(L),j} \leq b_j \tag{31}$$

$$\mathcal{V}(L) \cap \{j\} = \emptyset. \tag{32}$$

Inequality (31) ensures time window feasibility while inequality (32) ensures that the path is elementary.

If extension along the arc $(\eta(L), j)$ is feasible then a new label L' is created at node j. The information in label L' is set as follows:

$$\eta(L') = j \tag{33}$$

$$t(L') = \max\{a_j, t(L) + t_{\eta(L),j}\} \tag{34}$$

$$c(L') = c(L) + d_{\eta(L),j} \tag{35}$$

$$\mathcal{V}(L') = \mathcal{V}(L) \cup \{j\}. \tag{36}$$

Equation (33) sets the node of the new label, equation (34) sets the start time at the new label, equation (35) sets the cost of the new label, and equation (36) updates the set of visited nodes.

3.4.2 Dominance criterion

The following dominance criterion is used: a label L_1 dominates a label L_2 if

$$\eta(L_1) = \eta(L_2), \quad t(L_1) \leq t(L_2), \quad c(L_1) \leq c(L_2), \quad \mathcal{V}(L_1) \subseteq \mathcal{V}(L_2).$$

By defining $U(L) = \mathcal{V}(L) \cup \{i \in \{1, \ldots, 2n\} : t(L) + t_{\eta(L),i} > b_i\}$ be the set of *unreachable* nodes, the dominance criterion can be improved to

$$\eta(L_1) = \eta(L_2), \quad t(L_1) \leq t(L_2), \quad c(L_1) \leq c(L_2), \quad U(L_1) \subseteq U(L_2).$$

This idea was proposed by Feillet et al. (2004) and is similar to the improvement for the ESPPTWCPD described in Section 3.2.2.

We should point out that in the PDPTW, the load of a vehicle is not monotonous increasing or decreasing along a path as is the case for the VRP and VRPTW. As a result, including capacity constraints makes the pricing subproblem much harder to solve. This is why we have chosen to consider the ESPPTW instead of the ESPPTWC which was considered by Chabrier (2003) and Feillet et al. (2004).

3.5 ESPPTWP - SP4

The *Elementary Shortest Path Problem with Time Windows and Precedence Constraints* (ESPPTWP), denoted SP4, relaxes SP1 by removing the capacity and pairing constraints. In this relaxation, a path may visit the pickup node of a given vertex without visiting the corresponding delivery node, and vice-versa. However, if both nodes are visited, then the pickup node must be visited before the delivery node.

This shortest path algorithm is easily obtained from the algorithm for SP3 that uses the set U of unreachable nodes instead of $\mathcal{V}$. The definition of U in SP4 is:

$$U(L) = \mathcal{V}(L) \cup \{i \in \{1, \ldots, 2n\} : t(L) + t_{\eta(L),i} > b_i \vee i \notin \bigcap_{j \in \mathcal{V}(L)} \mathcal{S}_j\}$$

where $\mathcal{S}_j$ is the set of possible successors to node j. The basic definition of $\mathcal{S}_j$ is

$$\mathcal{S}_j = \begin{cases} (P \setminus \{j\}) \cup D \cup \{2n+1\} & \text{if } j \in P \\ (P \setminus \{j-n\}) \cup (D \setminus \{j\}) \cup \{2n+1\} & \text{if } j \in D. \end{cases}$$

If j is a delivery node then $j - n$ clearly is not a valid successor. It is possible to reduce the sets $\mathcal{S}_j$ even more. For example, nodes that cannot be visited after node j because of conflicting time windows can be removed from the set. This will not, however, have any impact on the lower bound obtained from the LPM.

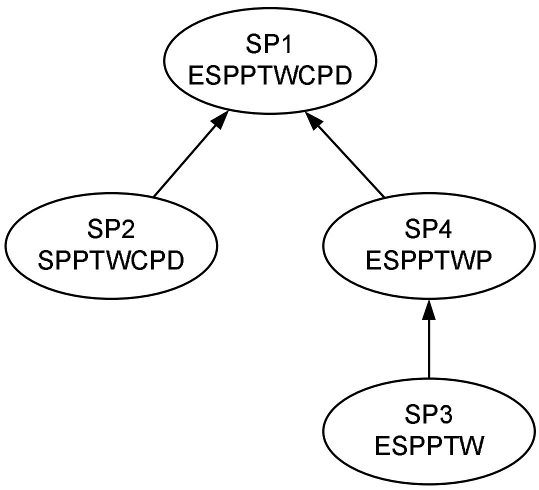
One may instead use the pairing and precedence constraints from the PDPTW to further reduce the set $\mathcal{S}_j$. For example, if j and i are two pickup nodes for which time windows and capacity constraints make it impossible to visit both i, $n+i$ and $n+j$ after j, then i and $n+i$ can be removed from $\mathcal{S}_j$ even though it may be possible to visit either i or $n+i$ after j if they are considered as individual nodes. These reductions may improve the lower bound obtained from the LPM because they transfer some of the pairing and precedence structure from the PDPTW to the ESPPTW.

3.6 Implementation issues

When implementing the labeling algorithms, one can store the sets $\mathcal{V}$ and $\mathcal{O}$ as binary vectors encoded as vectors of integers. In this representation, the i-th bit indicates whether the i-th request is part of the set. Consider for example a computer with 32-bit integers. If $n = 50$ then $\lceil \frac{50}{32} \rceil = 2$ integers are necessary to store each set. The labeling algorithms must perform a large number of dominance checks and each check requires that a set inclusion test be carried out twice (in case of SP1). The set inclusion test can be implemented using bitwise operations on the integers such that w bits can be processed in parallel, where w is the size of a machine word. To compare two words x and y one perform the operation $x \,\&\, y$, where & performs bitwise "and". If $x - (x \,\&\, y) = 0$ then the set corresponding to x is included in the set corresponding to y. The other inclusion can be tested in the same way. If both $x - (x \,\&\, y) \neq 0$ and $y - (x \,\&\, y) \neq 0$ then neither set is a subset of the other. To test for inclusion when the sets contain more than w items the operation is repeated for each word in the vector. This approach yields a significant speed improvement when compared to testing each bit separately. It has likely been used before but we are not aware of it having been described in the shortest path literature.

3.7 Strength of lower bounds using various pricing algorithms

Figure 3.7 shows the relative strength of the LP relaxation obtained using the different pricing algorithms. An arrow from X to Y indicates that the LP relaxation obtained by using pricing problem Y is tighter than the one obtained with pricing problem X.



3.8 Possible improvements

Irnich and Villeneuve (2003) have proposed a labeling algorithm that solves non-elementary shortest path problems while ensuring that cycles of length k or smaller do not occur. Their approach could be used to strengthen the lower bound of the LPM when using SP2 as a pricing problem since SP2 allows cycles containing more than two arcs. However, the computational results presented in section 6 show that the lower bound obtained with SP2 already are quite close to the lower bounds obtained with SP1, so it is not clear if the effort involved with forbidding longer cycles is worthwhile.

On a different note, Righini and Salani (2004) have proposed a bi-directional approach to shortest path problems with resource constraints. Instead of starting the label extension only from the source node, they simultaneously extend labels both from the source and the sink nodes. The two searches eventually meet at a point where the paths from the source are merged with paths from the sink. This approach has shown great potential for reducing the running time of the shortest path algorithm. It is out of the scope of the current paper to apply this technique to the shortest path problems considered, but it would be a promising area for future research.

4 Valid Inequalities

This section describes families of valid inequalities that can be used to strengthen the linear relaxation of the set partitioning formulation of the problem. To describe these inequalities, it is convenient to introduce new notation. For any node subset $S \subseteq V$, let $\delta^+(S) = \{(i,j) \in A | i \in S, j \notin S\}$. We also use $\delta^+(i)$ to designate the set $\delta^+(\{i\})$. Finally, let $x_{ij} = \sum_{k \in K} x_{ij}^k$.

4.1 Infeasible path inequalities

Cordeau (2005) and Ropke et al. (2005) discussed infeasible path inequalities and various strengthenings for the PDPTW. In this paper we will use two types of infeasible path inequalities. Consider an infeasible path $R = (k_1, \ldots, k_r)$, then the inequality

$$\sum_{i=1}^{r-1} x_{k_i,k_{i+1}} \leq r - 2 \tag{37}$$

is valid. In this paper the inequality is used as a simple way of handling ride time constraints, it is not believed to be very strong. Cordeau (2005) observed that the inequality can be strengthened if $k_1 = i$ and $k_r = n + i$ for some $i \in P$ and the path is infeasible because of time windows or ride time constraints. In that case the inequality can be strengthened to

$$\sum_{i=1}^{r-1} x_{k_i,k_{i+1}} \leq r - 3 \tag{38}$$

4.2 Fork inequalities

Let $R = (k_1, \ldots, k_r)$ be a feasible path in G and $S, T_1, \ldots, T_r \subset (P \cup D) \setminus R$ be subsets such that for any integer $h \leq r$ and any node pair $i \in S, j \in T_h$, the path $(i, k_1, \ldots, k_h, j)$ is infeasible. The following inequality is then valid for the PDPTW:

$$\sum_{i \in S} x_{i,k_1} + \sum_{h=1}^{r-1} x_{k_h,k_{h+1}} + \sum_{h=1}^{r} \sum_{j \in T_h} x_{k_h,j} \leq r. \tag{39}$$

Similarly, if $R = (k_1, \ldots, k_r)$ is a feasible path in G and $S_1, \ldots, S_r, T \subset (P \cup D) \setminus R$ are subsets such that for any integer $h \leq r$ and any node pair $i \in S_h, j \in T$, the path

$(i, k_h, \ldots, k_r, j)$ is infeasible, then the following inequality is valid for the PDPTW:

$$\sum_{h=1}^{r} \sum_{i \in S_h} x_{i,k_h} + \sum_{h=1}^{r-1} x_{k_h,k_{h+1}} + \sum_{j \in T} x_{k_r,j} \leq r. \tag{40}$$

Inequalities (39) and (40) were introduced by Ropke et al. (2005) and are called *outfork* and *infork inequalities*, respectively.

4.3 Reachability inequalities

For any node $i \in N$, let $A_i^- \subset A$ be the minimum arc set such that any feasible path from the origin depot 0 to node i uses only arcs from A_i^-. Let also A_i^+ be the minimum arc set such that any feasible path from i to the destination depot $2n+1$ uses only arcs in A_i^+. Consider a node set T such that each node in T must be visited by a different vehicle. This set is said to be *conflicting*. For any conflicting node set T, define the *reaching arc set* $A_T^- = \cup_{i \in T} A_i^-$ and the *reachable arc set* $A_T^+ = \cup_{i \in T} A_i^+$. For any node set $S \subseteq P \cup D$ and any conflicting node set $T \subseteq S$, the following two valid inequalities were introduced by Lysgaard (2005) for the VRP with time windows:

$$x(\delta^-(S) \cap A_T^-) \geq |T| \tag{41}$$

$$x(\delta^+(S) \cap A_T^+) \geq |T|. \tag{42}$$

These inequalities are obviously also valid for the PDPTW. In this problem, however, nodes can be conflicting not only because of time windows but also because of the precedence relationships and the capacity constraints. In the case of the DARP, the ride time constraints should also be taken into account when checking whether a pair of requests is conflicting.

4.4 Rounded capacity inequalities

Rounded capacity inequalities often used in the context of the vehicle routing problem (see, e.g., Naddef and Rinaldi, 2002) can also be used for the PDPTW. For any node subset $S \subseteq V \setminus (\{0, 2n+1\})$, the following inequality is valid:

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq \left\lceil \frac{\left|\sum_{i \in S} q_i\right|}{Q} \right\rceil.$$

4.5 Precedence inequalities

Let S be a subset of $V \setminus (\{0, 2n+1\})$ such that $i \in S$ and $n+i \notin S$ for some $i \in P$. Then the following inequality is valid:

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq 1.$$

Precedence inequalities were introduced by Ruland and Rodin (1997) in the context of the TSP with pickup and delivery.

4.6 Strengthened precedence cuts

Using ideas from the reachability inequalities, the precedence inequalities can be strengthened as follows. Let A_i be the set of arcs that can be used in a feasible path from i to $n+i$. Furthermore let S be a subset of $V \setminus (\{0, 2n+1\})$ such that $i \in S$ and $n+i \notin S$ for some $i \in P$. Then the following inequality is clearly valid for the PDPTW:

$$\sum_{(i,j)\in(\delta^+(S)\cap A_i)} x_{ij} \geq 1.$$

5 Branch-and-Cut-and-Price Algorithm

5.1 Overview

Branch-and-cut is a well-known solution paradigm which has proved to very efficient for the solution of several families of combinatorial problems. In particular, this approach has been successful in solving the TSP and the VRP. In branch-and-cut algorithms, some constraints are relaxed and introduced dynamically in the model when they are violated by the solution to the relaxation solved in a node of the branch-and-bound tree. This is accomplished by solving a separation problem to identify violated inequalities. Branch-and-cut-and-price is a variant of branch-and-cut in which the linear programming relaxations obtained at each node of the branch-and-bound tree are solved by column generation.

It is well known that the running time of branch-and-price algorithms can be improved by using heuristic algorithms for the pricing problem. As long as the heuristic algorithms are able to find columns with negative reduced-cost one can add those columns to the LPM and solve the problem again. If the heuristics fail to identify columns with a negative reduced-cost, one then has to apply an exact pricing algorithm to verify whether no negative reduced-cost columns exist, or to find one or more columns that can be added to the LPM.

Ideally it should be necessary to call the exact pricing algorithm only once for each node in the branch-and-bound tree to verify that no reduced-cost column exists. In fact, this is not even necessary if the relaxation value associated with a node is lower than the current upper bound. In this case, the lower bound will not be used to fathom the node and it is not necessary to find the optimal relaxation value for this node. To use this strategy, however, one needs good pricing heuristics to avoid branching on incorrect data.

Every time the LPM has been solved we are faced with a choice of either trying to generate more variables (columns) or valid inequalities (rows). Adding both at the same time does not seem to make sense as the inclusion of more violated valid inequalities probably implies that other variables are needed compared to the ones we need with the current model. The simplest approach would be to add variables until no more variables with negative reduced cost exists and then try to generate violated inequalities.

We take a very similar approach: variables are generated as long as the heuristics are able to identify promising variables. When the heuristics fail, the cut generation routines take over unless the lower bound is above the upper bound or if cut generation has been tried before without success in the current branch and bound node. If cut generation is tried and identifies violated inequalities, then they are added to the model and the pricing heuristics are allowed to try to find variables with negative reduced costs again. If the cut generation is unsuccessful or if it is not tried because of the reasons listed above then the exact pricing algorithm is called.

In the following sections, we first describe preprocessing techniques to reduce the size of the problem and strengthen some of its parameters. We then explain the branching strategy used to explore the enumeration tree, and the separation procedures for the identification of violated valid inequalities. We finally describe several heuristics which can be used to solve the pricing problem.

5.2 Preprocessing

Several preprocessing rules for tightening time windows in the PDPTW and the DARP have been described in Dumas et al. (1991) and Cordeau (2005). All these rules have been implemented here. In addition, Desrochers et al. (1992) propose the following four rules for tightening time windows in the VRPTW:

1. Minimal arrival time from predecessors

$$a_k = \max\left\{a_k, \min\left\{b_k, \min_{(i,k)\in A'}\{a_i + t_{ik}\}\right\}\right\}$$

2. Minimal arrival time to successors

$$a_k = \max\left\{a_k, \min\left\{b_k, \min_{(k,j)\in A'}\{a_j - t_{kj}\}\right\}\right\}$$

3. Maximal departure time from predecessors

$$b_k = \min\left\{b_k, \max\left\{a_k, \max_{(i,k)\in A'}\{b_i + t_{ik}\}\right\}\right\}$$

4. Maximal departure time to successors

$$b_k = \min\left\{b_k, \max\left\{a_k, \max_{(k,j)\in A'}\{b_j - t_{kj}\}\right\}\right\}$$

In these rules, the set A' is the set of feasible arcs for the problem. Cordeau (2005) describes how to compute this set from A. The four rules are applied to each node in a cyclic fashion. In combination with the rules described in Dumas et al. (1991) and Cordeau (2005) one actually need only apply rules 2 and 3.

In the case of the DARP, however, care must be taken when applying these time window tightening rules. Indeed, one cannot tighten the start of the time window of a delivery node or the end of the time window of a pickup node as this may then lead to an increase of the ride time associated to the corresponding request.

5.3 Branching strategy

When the solution of the LPM is fractional and no violated inequality can be identified, one has to resort to branching. Branching in a column generation algorithm should be done with care as the branching strategy should preferably be compatible with the algorithm used for solving the pricing problem, i.e., the same type of pricing problem should be solved in the child nodes as in the parent node. This implies that branching decisions should be easily transferred to the subproblem and should not change its structure.

Three branching strategies have been implemented which are compatible with the pricing problems considered in this paper except for solving SP1 and SP2 using dominance criteria (DOM1') and (DOM2).

The first strategy branches on the arc variables x_{ij}. The procedure for doing this is described in detail by Desrochers et al. (1992). Our implementation branches on the arc with a flow closest to 0.5.

The second strategy branches on the outflow of a set of nodes as proposed for VRP by Naddef and Rinaldi (2002). A set of nodes S is first selected such that $x(\delta^+(S))$ is as fractional as possible. Two branches are then created: $x(\delta^+(S)) \leq \lfloor x(\delta^+(S)) \rfloor$ and $x(\delta^+(S)) \geq \lceil x(\delta^+(S)) \rceil$. In our implementation, the set S is found using a simple greedy heuristic.

The last strategy calculates $x(\delta^+(0))$. If $x(\delta^+(0))$ is fractional then the two branches $x(\delta^+(0)) \leq \lfloor x(\delta^+(0)) \rfloor$ and $x(\delta^+(0)) \geq \lceil x(\delta^+(0)) \rceil$ are created. If $x(\delta^+(0))$ is integer then one of the two previous branching strategies is used. This rule was proposed by Desrochers et al. (1992). This branching strategy is often called *branching on the number of vehicles.*

In our branch-and-cut-and-price algorithm, the enumeration tree is explored in a depth-first fashion. This choice is motivated by the availability of high-quality heuristics for the PDPTW which generally provide tight upper bounds.

5.4 Separation routines

We refer the reader to Ropke et al. (2005) for a description of the separation procedures for the fork, capacity, reachability and precedence inequalities. An exact, polynomial-time separation procedure for the strengthened precedence inequality is described below.

5.4.1 Strengthened precedence inequality

Before starting the branch-and-bound procedure the sets A_i are calculated for every request i. If the sets A_i^- and A_i^+ are known then one can use the fact that $A_i \subseteq A_{n+i}^- \cap A_i^+$ to speed up the calculation of A_i. In order to separate the inequality the following procedure is applied for every request i:

1. Construct a graph with nodes $\{1, \ldots, 2n\}$ and arcs A_i.
2. Set the the weight of each arc (i, j) equal to $\tilde{x}_{ij}$ where $\tilde{x}_{ij}$ is the total flow on arc (i, j) in the current solution to LPM.
3. Solve a minimum cut problem on the graph with node i as source and node $n + i$ as sink, yielding a set $S \ni i$.

If the flow across the minimum cut is less than 1, then one has identified a violated inequality $\sum_{(i,j)\in(\delta^+(S)\cap A_i)} x_{ij} \geq 1$.

5.4.2 Cut pool management

Every time the LPM is solved the algorithm determines which of the valid inequalities previously generated are satisfied at equality by the current solution. If an inequality has not been binding for ten consecutive iterations, it is removed from the problem and inserted in a cut pool. Every time the LPM is solved, the cut pool is checked for violated inequalities. If a violated inequality is found in the cut pool, it is then added to the linear relaxation and the problem is solved again. Once an inequality has been identified by one of the separation procedures it will always stay in the program, either explicitly in the model, or implicitly in the cut pool. In computational experiments, a significant performance boost was observed when using this approach compared to keeping all inequalities in the formulation.

5.5 Pricing problem heuristics

We first present in Section 5.5.1 two general heuristics which are valid for all the pricing problems described in section 3. They both work by truncating the labeling algorithms. In Section 5.5.2 and 5.5.3 we then introduce more specialized heuristics that work by the classic construction and improvement principle.

5.5.1 Label heuristics

It has previously been proposed to turn exact labeling algorithms into heuristics by limiting the number of labels created in different ways. For example, Dumas et al. (1991) proposed to reduce the network before running the pricing algorithm. They created networks with between 30% and 50% of the best arcs. Irnich and Villeneuve (2003) used a variant of this idea by creating reduced networks G_l where each node is connected to its l nearest neighbors.

In this paper we let the reduced network G_l consist of the shortest arcs with respect to d_{ij}. Each node $i \in \{1, \dots, 2n\}$ is incident with at most l outgoing arcs reaching a pickup node and l outgoing arcs reaching a delivery node. After construction, all feasible arcs $(i, n+i), (0, i), (n+i, 2n+1), i \in \{0, \dots, n\}$ which are not already in G_l are added to G_l. The arcs are grouped into pickup/delivery arcs to keep the network balanced. The current implementation uses two reduced networks G_5 and G_{10}. If the search using network G_5 does not find any negative paths then it switches to network G_{10}. The corresponding heuristics is denoted $H1$.

Dumitrescu (2002) proposed to limit the number of unprocessed labels at any time. For our purpose this corresponds to putting a limit on U ($|U| \leq \mu$) in Algorithm 1. Only the μ best (with respect to the reduced cost) labels are kept, the worst labels being discarded. This heuristic is used in a three-phase fashion. First a limit $\mu = 500$ is used. If the heuristic does not find any negative cost paths then $\mu = 1000$, and finally $\mu = 2000$ is tried. This heuristic is denoted $H2$.

If $H2$ does not return any negative cost paths then, it can in some cases be proved that no negative cost paths exists. This happens if one has never discarded labels because of the limit on unprocessed labels. However, this is only likely to happens for easy problem instances. A similar property was observed by Jepsen et al. (2005) when creating a heuristic by limiting the total number of labels processed.

For both of the heuristics and the exact algorithms based on the labeling algorithm we generate more than one negative cost column, but stop the algorithm if 100 different negative reduced-cost columns have been generated. All negative reduced cost columns are added to the LPM.

5.5.2 Construction heuristic

Heuristics that do not use the label setting algorithm have been implemented to determine whether another heuristic paradigm could provide the same or better solution quality as the heuristics based on labeling algorithms which are the most popular in column generation algorithms.

Sol (1994) proposed to use a cheapest insertion heuristic to solve the ESPPTWCPD. A similar approach is implemented in this paper. Starting from a route containing only request i we add the request that increases the least the reduced-cost of the path. During insertions we keep track of the best route observed. The process is repeated with every request as a starting point. This algorithm is denoted $H3$.

A straightforward way to improve this heuristic is by randomizing it. This can be done by performing insertions that are not the most promising: the possible insertions are ranked by insertion cost and a request is chosen by a random process that tends to select insertions with low cost. When using the randomized insertion it is worthwhile to try to construct a route starting from the same initial route containing request i several times. This algorithm is denoted $H4$.

5.5.3 LNS heuristic

It is well known that improvement or steepest descent heuristics often produce high quality solutions in little time. This has been used by Savelsbergh and Sol (1998) (see also Sol (1994)) to propose improvement heuristics for the pricing problem. As an initial solution, routes from the current LPM with reduced cost 0 were used. Their neighborhood move consisted of removing one node and inserting another. Notice that this move never changes the length of the initial path.

In this section we describe a different improvement heuristic which is based on the Large Neighborhood Search (LNS) introduced by Shaw (1998). Ropke and Pisinger (2004) showed that the LNS can be easily implemented by using simple construction heuristics, an idea that will be used here. The LNS algorithm attempts to improve an initial path by alternating between removing requests from the path and inserting requests into the path. The requests to remove are chosen randomly and requests are inserted using the randomized insertion algorithm outlined in section 5.5.2.

The pseudo-code for the LNS is shown in Algorithm 2. The algorithm takes a path p and an integer σ as input. The parameter σ determines how many times the removal/insertion iterations should be performed without improving the path.

In line 3 we set f, the cost of the currently best solution. Line 4 makes the algorithm continue as long as an improvement is found. In line 6 nodes are removed from the path. The function `removeNodes`(p) returns a path where up to 50% of the nodes from p have been removed, p itself is not changed. In line 7 nodes are inserted into the path again, the

Algorithm 2 LNS pseudo code.

```
1  Input:  Path p, integer σ
2  for i = 1,...,σ
3    f = ∞;
4    while (c(p) < f)
5      f = c(p);
6      p' = removeNodes(p);
7      p' = randomizedInsert(p');
8      if c(p') < f
9        p = p';
10 return p;
```

insertion is randomized but good insertions are favored. Any unplanned node can be inserted. In line 8 and 9 the current solution is updated if an improvement was found. The functions `removeNodes` and `randomizedInsert` are dependent on the shortest path problem that is solved (e.g. they take into account whether the path must satisfy the pairing constraint).

The improvement heuristic is used in two contexts: In heuristic H5, LNS is used to improve the paths that are selected ($y_r > 0$) in the current LP solution, in H5, σ is set to 20. In heuristic H6, LNS is used to improve the paths generated by the randomized insertion heuristic described in Section 5.5.2. The LNS heuristic is applied to paths with reduced cost greater than or equal to zero to try to bring the reduced-cost below 0. In H6, σ is set to 5 as many paths are given to the improvement heuristic.

6 Computational experiments

This section describes the computational experiments that we have performed on several sets of test instances for both the PDPTW and the DARP. The algorithm was implemented in C++ and all experiments were carried out on an AMD Opteron 250 computer (2.4 GHz) running Linux. CPLEX 9.0 was used as LP solver and the COIN-OR Open Solver Interface (OSI, `http://www.coin-or.org/index.html`) was used as an interface to the LP solver. In all experiments, a limit of two hours of CPU time was used unless otherwise indicated.

For the PDPTW, we have used two main sets of instances. The first one was introduced by Li and Lim (2001) and is based on the well-known Solomon test problems for the VRPTW.

The second set of instances was introduced by Ropke et al. (2005) and is based on a generator initially proposed by Savelsbergh and Sol (1998). As explained by Ropke et al., the generator was modified to obtain harder instances by reducing the ratio between the travel times and the length of the planning horizon. In addition, the new generator considers a single depot located at the middle of a square instead of a different depot for each vehicle.

In all instances, the coordinates of each pickup and delivery location are chosen randomly according to a uniform distribution over the $[0, 50] \times [0, 50]$ square. The load q_i of request i is selected randomly from the interval $[5, Q]$, where Q is the vehicle capacity. A planning horizon of length $T = 600$ is considered and each time window has width W. The time windows for request i are constructed by first randomly selecting e_i in the interval $[0, T - t_{i,n+i}]$ and then setting $l_i = e_i + W$, $e_{n+i} = e_i + t_{i,n+i}$ and $l_{n+i} = e_{n+i} + W$. In all instances, the primary objective consists of minimizing the number of vehicles, and a fixed cost of 10^4 is thus imposed on each outgoing arc from the depot.

Five groups of instances were generated by considering different values of Q and W. The characteristics of these groups are summarized in Table 4. Ropke et al. (2005) considered ten instances with $30 \leq n \leq 75$ in each of the first four groups. Here, we introduce larger instances with $100 \leq n \leq 200$ as well as a new group of instances (group E) with $Q = 30$. This yields a total of 75 instances. The name of each instance (e.g., A50) indicates the class to which it belongs and the number of requests it contains.

Table 4: Characteristics of the new PDPTW instances

Class	Q	W
A	15	60
B	20	60
C	15	120
D	20	120
E	30	120

The third set of instances that we have used for testing was introduced by Cordeau (2005) for the DARP. These consist of randomly generated Euclidean DARP instances comprising up to 96 requests. They all have narrow time windows of 15 minutes. In the first subset ('a' instances), $q_i = 1$ for every request i and the vehicle capacity is $Q = 3$. In the second set ('b' instances), q_i belongs to the interval $[1, 6]$ and $Q = 6$. These data are described in detail in Cordeau (2005) and are available on the following web site: `http://www.hec.ca/chairedistributique/data/darp`. Their main characteristics are summarized in Table 5. In this table, columns $|K|$ and T indicate, respectively, the number of available vehicles and the length of the planning horizon in which time windows are generated. The constraint on the number of vehicles is easily imposed in our formulations as a bound on the total outgoing flow from the origin depot.

The pricing algorithms operate on travel times with a fixed number of decimals. Thus for the DARP and Li and Lim instances distances and travel times have been truncated (rounded down) to four decimals. For the PDPTW instances similar to the ones proposed by Savelsbergh and Sol, distances and travel times have been rounded up to two decimals. We have less precision for these instances to avoid numerical problems due to the fixed costs on vehicles that results in high route costs. Travel times are rounded up to ensure that the travel times satisfy the triangle inequality. For the DARP and Li and Lim instances this is not a problem as a service time is associated with each request.

Our computational experiments focus on four aspects. First, we wished to investigate the impact of the various subproblems described in Section 3. Second, we wanted to measure the impact of the valid inequalities described in Section 4 on the performance of the branch-and-price algorithm. Third, we wanted to compare the performance of our branch-and-cut-and-price algorithm to the branch-and-cut algorithm of Ropke et al. (2005). Fourth, we wanted to measure the impact of the pricing heuristics.

Table 5: Characteristics of DARP instances

Instance	$\lvert K\rvert$	n	T	Q	L	Instance	$\lvert K\rvert$	n	T	Q	L
a2-16	2	16	480	3	30	b2-16	2	16	480	6	45
a2-20	2	20	600	3	30	b2-20	2	20	600	6	45
a2-24	2	24	720	3	30	b2-24	2	24	720	6	45
a3-24	3	24	480	3	30	b3-24	3	24	480	6	45
a3-30	3	30	600	3	30	b3-30	3	30	600	6	45
a3-36	3	36	720	3	30	b3-36	3	36	720	6	45
a4-32	4	32	480	3	30	b4-32	4	32	480	6	45
a4-40	4	40	600	3	30	b4-40	4	40	600	6	45
a4-48	4	48	720	3	30	b4-48	4	48	720	6	45
a5-40	5	40	480	3	30	b5-40	5	40	480	6	45
a5-50	5	50	600	3	30	b5-50	5	50	600	6	45
a5-60	5	60	720	3	30	b5-60	5	60	720	6	45
a6-48	6	48	480	3	30	b6-48	6	48	480	6	45
a6-60	6	60	600	3	30	b6-60	6	60	600	6	45
a6-72	6	72	720	3	30	b6-72	6	72	720	6	45
a7-56	7	56	480	3	30	b7-56	7	56	480	6	45
a7-70	7	70	600	3	30	b7-70	7	70	600	6	45
a7-84	7	84	720	3	30	b7-84	7	84	720	6	45
a8-64	8	64	480	3	30	b8-64	8	64	480	6	45
a8-80	8	80	600	3	30	b8-80	8	80	600	6	45
a8-96	8	96	720	3	30	b8-96	8	96	720	6	45

6.1 Pricing algorithms

Six different pricing algorithms will be used in the rest of this section. The SP1 pricing problem is solved using two algorithms. The algorithm denoted SP1* uses the algorithm based on the (DOM1') dominance criterion, while the algorithm denoted SP1 uses the algorithm based on the (DOM1†) dominance criterion. Similarly SP2 uses the (DOM2†) criterion while SP2* uses the (DOM2) criterion. The (DOM1') and (DOM2) criteria are stronger than the (DOM1†) and DOM2†) criteria, but they are not compatible with our branching scheme and cutting planes (see Sections 3.2 and 3.3) so the SP1* and SP2* algorithms are only used to calculate a lower bound.

For the SP3 and SP4 pricing problems we only have one pricing algorithm for each problem.

Notice that *SP1* and *SP2* are used to denote both a PDPTW relaxation and an algorithm. The meaning should be clear from the context.

6.2 Pricing heuristics

In this section, we report the results of experiments performed with the pricing heuristics introduced in Section 5.5. The heuristics have been tested on series 1 of the 50 request test set proposed by Li and Lim (2001) as these instances turn out to produce hard pricing problems.

To limit the number of tables, we only report results for relaxation SP1. We propose a number of configurations of the pricing heuristic and test how long it takes to prove the lower bound for each instance when using the particular heuristic configuration together with either exact algorithm SP1* or SP1. Results for SP1* are shown in Table 7 while results for SP1 are shown in Table 8.

Nine combinations (A2–A10) of the heuristics were tested and the algorithm was also tested without any heuristic (A1). Table 6 gives an overview of these configurations. The left column shows the configuration name and the right one shows the heuristics used in the configuration. The sequence of the heuristics shows their calling sequence. As an example, in configuration A3, heuristic H3 is tried first and if this fails to find paths with negative reduced cost then heuristic H1 is tried. If this also fails then the algorithm resorts to the exact shortest path algorithm.

Only the construction heuristic (H3) was tested alone (configuration A2). The rest of the heuristics were tested together with the construction heuristic as it quickly can produce some routess early in the column generation process. Configurations A3 to A7 test one heuristic together with the construction heuristic. Configuration A8 tests the two heuristics based on the label setting algorithm together. Configuration A9 tests the randomized insertion together with the LNS heuristics and configuration A10 includes all heuristics.

Configuration name	Heuristics and sequence
A1	None
A2	H3
A3	H3-H1
A4	H3-H2
A5	H3-H4
A6	H3-H5
A7	H3-H6
A8	H3-H2-H1
A9	H3-H4-H5-H6
A10	H3-H4-H5-H6-H2-H1

Table 6: Overview of pricing heuristic configurations.

Table 7 and 8 show the results of the tests. For every heuristic configuration the table contains three columns: *ok* - indicates if the lower bound in the root node was proved within the time limit (2 hours), *time (s)* - the total time needed to prove the lower bound in the root node (in seconds). *#ex* - number of calls to the exact pricing algorithm needed to prove the lower bound. Blank entries indicate that a time out occurred. The row *Sum* sums each column (in case of the *ok* column it counts the number of proved lower bounds) while the *Sum'* column sums over the instances that could be solved by all configurations.

The results clearly show the importance of using good pricing heuristics. For the SP1* algorithm we can observe speedups by more than a factor 250 between having no pricing heuristic and using the heuristics given by configuration A10 (see instance lr104), for the SP1 algorithm the highest speedup factor is 60 (instance lrc104, A1 vs. A10). Using even the simplest construction heuristic helps significantly as the first pricing problems are especially hard to solve using the exact algorithm as many negative cycles exists.

It can be seen that the configurations that use the LNS pricing heuristics (heuristics H5 and H6, configurations A6, A7, A9 and A10) are performing well, so using more advanced local search heuristics to solve pricing problems is a worthwhile research path. For the SP1 algorithm it can be seen that the configurations that are able to prove the lower bound of lc109 all use heuristic H1, so the more traditional pricing heuristics are still useful. It is clear that configuration A10 that uses all heuristics is the most powerful. If one looks at the number of calls to the exact pricing algorithm when using A10 one sees that the heuristic is close to reducing the number of calls to 1 which is what we can hope for (the exact pricing algorithm is called at the end to prove that no negative cost paths exist). Notice that entries where 0 calls to the exact algorithm were carried out are the ones where optimality is proved by heuristic H2 (see Section 5.5.1).

It is also interesting to note that reducing the number of calls to the exact pricing algorithm from 10 down to 2 (for example) often will result in a speed up larger than 5, as the pricing problem tends to get easier towards the end of the column generation process. The two tables also show that SP1* is much more powerful than SP1. This finding will be confirmed in the following tests.

Configuration A10 will be used in the following tests unless otherwise noted.

	A1			A2			A3			A4			A5		
	ok	time (s)	#ex	ok	time (s)	#ex	ok	time (s)	#ex	ok	time (s)	#ex	ok	time (s)	#ex
lr101	X	0.3	16	X	0.2	1	X	0.2	1	X	0.2	0	X	0.2	1
lr102	X	2.3	28	X	0.3	3	X	0.3	2	X	0.6	1	X	0.3	1
lr103	X	74.3	34	X	1.2	4	X	0.7	1	X	1.8	2	X	1.4	4
lr104	X	2479.6	39	X	60.2	10	X	30.3	5	X	101.4	11	X	64.4	8
lr105	X	0.3	22	X	0.1	2	X	0.2	1	X	0.1	0	X	0.2	1
lr106	X	1.1	27	X	0.2	3	X	0.3	2	X	0.5	1	X	0.4	3
lr107	X	930.8	40	X	15.2	11	X	2.8	3	X	8.2	6	X	16.4	6
lr108				X	5390.4	14	X	1004.2	4	X	1843.2	16	X	1236.3	16
lr109	X	3.5	41	X	1.0	9	X	0.8	2	X	1.7	3	X	1.2	5
lr110	X	35.9	37	X	7.1	9	X	5.4	4	X	10.7	10	X	4.5	5
lr111	X	47.5	35	X	3.1	9	X	2.2	3	X	5.4	4	X	1.8	4
lr112				X	4163.6	18	X	1072.1	6	X	3285.0	15	X	2535.1	12
lc101	X	0.4	35	X	0.1	1	X	0.1	1	X	0.1	0	X	0.2	1
lc102	X	27.9	33	X	0.3	3	X	0.3	1	X	0.7	1	X	0.5	2
lc103				X	8.7	8	X	7.2	5	X	15.6	5	X	7.3	6
lc104															
lc105	X	1.2	33	X	0.1	1	X	0.1	1	X	0.2	0	X	0.2	1
lc106	X	2.8	32	X	0.3	4	X	0.3	1	X	0.8	1	X	0.5	3
lc107	X	3.9	36	X	0.3	3	X	0.4	2	X	0.5	1	X	0.4	1
lc108	X	28.7	44	X	1.1	9	X	1.1	3	X	3.8	2	X	3.3	8
lc109	X	224.6	47	X	5.4	13	X	3.4	4	X	9.6	6	X	6.5	10
lrc101	X	0.3	18	X	0.2	2	X	0.2	1	X	0.2	0	X	0.2	1
lrc102	X	2.4	28	X	0.4	4	X	0.5	2	X	0.9	1	X	0.5	2
lrc103	X	8.8	34	X	2.4	12	X	1.5	2	X	3.0	3	X	2.4	9
lrc104	X	3055.3	47	X	434.6	11	X	379.4	3	X	448.8	11	X	318.5	10
lrc105	X	0.8	29	X	0.3	6	X	0.4	2	X	0.4	0	X	0.4	3
lrc106	X	1.5	33	X	0.5	8	X	0.6	2	X	1.0	1	X	0.4	2
lrc107	X	9.7	33	X	3.1	10	X	3.2	3	X	4.3	5	X	2.9	7
lrc108	X	81.1	37	X	35.8	14	X	31.1	6	X	40.3	14	X	28.4	10
Sum	25	7024.5	838	28	10136.1	202	28	2549.2	73	28	5788.9	120	28	4234.6	142
Sum'		7024.5	838		573.5	162		465.7	58		645.1	84		456.0	108

	A6			A7			A8			A9			A10		
	ok	time (s)	#ex	ok	time (s)	#ex	ok	time (s)	#ex	ok	time (s)	#ex	ok	time (s)	#ex
lr101	X	0.2	1	X	0.3	1	X	0.2	0	X	0.3	1	X	0.3	0
lr102	X	0.3	1	X	0.4	1	X	0.7	1	X	0.5	1	X	0.8	1
lr103	X	1.4	4	X	1.8	4	X	1.8	1	X	2.0	4	X	2.4	1
lr104	X	34.5	8	X	19.9	3	X	41.0	5	X	10.3	2	X	8.6	1
lr105	X	0.2	1	X	0.3	1	X	0.1	0	X	0.3	1	X	0.2	0
lr106	X	0.3	2	X	0.5	2	X	0.5	1	X	0.5	1	X	0.8	1
lr107	X	9.7	6	X	3.3	2	X	6.2	2	X	9.2	3	X	9.1	2
lr108	X	525.3	11	X	282.3	4	X	303.0	5	X	196.0	5	X	94.4	2
lr109	X	0.8	6	X	0.8	1	X	1.9	1	X	1.2	1	X	1.6	1
lr110	X	4.0	4	X	3.3	3	X	6.5	2	X	1.9	1	X	2.6	1
lr111	X	1.5	4	X	2.5	3	X	6.5	3	X	2.9	4	X	3.0	1
lr112	X	965.0	6	X	681.6	3	X	1107.2	6	X	330.6	2	X	350.0	2
lc101	X	0.1	1	X	0.3	1	X	0.1	0	X	0.3	1	X	0.2	0
lc102	X	0.3	1	X	0.7	1	X	0.7	1	X	0.7	1	X	1.1	1
lc103	X	3.3	3	X	7.4	5	X	19.1	4	X	3.5	1	X	4.8	1
lc104													X	6763.7	3
lc105	X	0.1	1	X	0.3	1	X	0.2	0	X	0.4	1	X	0.3	0
lc106	X	0.4	4	X	0.8	2	X	0.8	1	X	1.3	2	X	1.6	1
lc107	X	0.4	3	X	0.8	1	X	0.6	1	X	0.8	1	X	1.1	1
lc108	X	0.8	3	X	2.8	4	X	4.0	1	X	4.3	3	X	4.9	1
lc109	X	3.6	6	X	5.5	3	X	11.7	4	X	7.8	4	X	9.8	1
lrc101	X	0.2	1	X	0.3	1	X	0.2	0	X	0.3	1	X	0.2	0
lrc102	X	0.4	2	X	0.4	1	X	1.0	1	X	0.7	1	X	1.0	1
lrc103	X	1.0	4	X	2.0	3	X	3.0	1	X	2.7	3	X	2.9	1
lrc104	X	233.6	6	X	257.5	4	X	445.1	3	X	174.7	4	X	102.1	2
lrc105	X	0.3	4	X	0.5	1	X	0.4	0	X	0.7	1	X	0.7	0
lrc106	X	0.4	4	X	0.6	1	X	1.1	1	X	0.7	2	X	0.9	1
lrc107	X	1.6	4	X	2.5	4	X	4.7	1	X	2.9	5	X	2.8	1
lrc108	X	13.6	6	X	20.2	7	X	21.3	3	X	16.9	6	X	13.2	2
Sum	28	1803.3	107	28	1299.5	68	28	1989.3	49	28	774.5	63	29	7385.1	30
Sum'		309.7	87		328.2	56		560.1	34		244.4	55		172.2	22

Table 7: Pricing problem heuristics used with exact pricing algorithm SP1*.

	A1			A2			A3			A4			A5		
	ok	time (s)	#ex	ok	time (s)	#ex	ok	time (s)	#ex	ok	time (s)	#ex	ok	time (s)	#ex
lr101	X	0.5	20	X	0.2	1	X	0.3	1	X	0.2	0	X	0.3	1
lr102	X	11.5	25	X	1.3	2	X	0.8	1	X	1.2	1	X	0.8	1
lr103	X	139.1	32	X	35.3	10	X	13.1	3	X	13.3	3	X	28.2	8
lr104				X	4725.1	12	X	2543.5	7	X	3784.9	10	X	2015.6	6
lr105	X	0.6	22	X	0.3	2	X	0.3	1	X	0.3	0	X	0.4	1
lr106	X	3.7	27	X	1.1	7	X	0.8	1	X	1.3	1	X	0.9	3
lr107	X	1814.8	40	X	502.9	11	X	188.5	4	X	551.1	11	X	320.9	7
lr108				X	2578.1	17	X	1466.3	7	X	2400.5	14	X	1879.0	14
lr109	X	5.7	36	X	2.0	9	X	1.2	1	X	2.1	2	X	2.0	6
lr110	X	72.0	33	X	24.9	11	X	16.5	6	X	16.7	6	X	7.5	3
lr111	X	599.3	38	X	100.5	8	X	27.3	2	X	43.7	3	X	111.8	9
lr112				X	6730.5	18	X	3288.2	9			18	X	5104.1	13
lc101	X	1.7	36	X	0.8	2	X	0.5	1	X	0.5	0	X	0.5	1
lc102				X	1020.7	3	X	348.8	1	X	374.8	1	X	670.8	2
lc103															
lc104															
lc105	X	5.8	41	X	0.9	1	X	0.7	1	X	0.9	1	X	0.7	1
lc106	X	165.7	33	X	34.6	6	X	14.0	2	X	14.4	2	X	37.0	6
lc107	X	119.6	37	X	13.5	4	X	4.9	1	X	5.1	1	X	4.1	1
lc108				X	2379.3	9	X	1326.7	5	X	869.4	3	X	1962.5	7
lc109							X	3514.7	2						
lrc101	X	0.7	20	X	0.5	3	X	0.3	1	X	0.3	0	X	0.6	2
lrc102	X	6.0	32	X	1.4	5	X	0.7	1	X	2.4	3	X	1.4	3
lrc103	X	22.0	26	X	9.3	12	X	2.6	1	X	7.0	5	X	4.6	5
lrc104	X	2737.6	47	X	588.9	13	X	227.1	4	X	716.0	13	X	296.0	7
lrc105	X	1.8	32	X	0.8	6	X	0.9	2	X	0.7	1	X	0.9	2
lrc106	X	2.2	29	X	0.9	5	X	1.1	3	X	0.9	1	X	1.0	3
lrc107	X	18.3	31	X	5.8	9	X	5.2	4	X	5.4	4	X	5.5	7
lrc108	X	178.4	42	X	62.8	13	X	42.2	6	X	64.7	12	X	43.0	8
Sum	21	5906.9	679	26	18822.4	199	27	13037.1	78	25	8877.9	116	26	12499.7	127
Sum'		5906.9	679		1388.8	140		549.0	47		1448.2	70		867.7	85

	A6			A7			A8			A9			A10		
	ok	time (s)	#ex	ok	time (s)	#ex	ok	time (s)	#ex	ok	time (s)	#ex	ok	time (s)	#ex
lr101	X	0.3	1	X	0.3	1	X	0.2	0	X	0.3	1	X	0.3	0
lr102	X	0.7	1	X	0.9	1	X	1.3	1	X	1.0	1	X	1.5	1
lr103	X	14.2	4	X	11.3	3	X	6.9	1	X	8.5	2	X	6.2	1
lr104	X	2614.8	6	X	704.1	2	X	3195.8	7	X	684.6	2	X	769.3	2
lr105	X	0.3	1	X	0.5	1	X	0.3	0	X	0.5	1	X	0.4	0
lr106	X	0.9	2	X	0.9	1	X	1.5	1	X	0.9	1	X	1.3	1
lr107	X	326.4	7	X	180.6	4	X	144.3	3	X	46.9	1	X	53.7	1
lr108	X	1452.0	10	X	977.5	5	X	715.7	6	X	870.6	5	X	499.9	2
lr109	X	1.4	4	X	1.7	2	X	1.9	1	X	1.4	1	X	1.8	1
lr110	X	7.3	3	X	5.7	2	X	11.7	3	X	3.5	1	X	4.5	1
lr111	X	64.5	5	X	62.7	5	X	32.2	2	X	51.2	4	X	18.3	1
lr112	X	2741.3	7	X	1434.4	4	X	2549.3	6	X	724.8	2	X	803.5	2
lc101	X	0.8	2	X	0.6	1	X	0.5	0	X	0.7	1	X	0.6	0
lc102	X	1012.6	3	X	337.4	1	X	370.2	1	X	337.9	1	X	344.5	1
lc103															
lc104															
lc105	X	0.6	1	X	0.8	1	X	1.0	1	X	0.9	1	X	1.3	1
lc106	X	30.5	5	X	18.4	3	X	8.9	1	X	13.7	2	X	9.4	1
lc107	X	10.6	3	X	4.3	1	X	5.9	1	X	4.2	1	X	5.5	1
lc108	X	1920.3	7	X	518.9	2	X	890.0	3	X	520.0	2	X	265.5	1
lc109							X	5813.0	3				X	1898.7	1
lrc101	X	0.6	2	X	0.5	1	X	0.3	0	X	0.5	1	X	0.5	0
lrc102	X	1.2	3	X	1.2	2	X	1.9	1	X	1.5	2	X	1.5	1
lrc103	X	3.5	4	X	4.2	4	X	5.2	1	X	2.9	2	X	3.4	1
lrc104	X	468.6	9	X	137.5	3	X	430.3	6	X	80.1	2	X	45.6	1
lrc105	X	0.8	4	X	0.6	1	X	0.7	1	X	0.7	1	X	1.0	1
lrc106	X	0.9	3	X	1.2	2	X	1.0	1	X	1.0	1	X	1.3	1
lrc107	X	3.8	5	X	3.4	3	X	5.5	3	X	3.5	3	X	3.3	1
lrc108	X	23.6	5	X	21.3	4	X	39.3	5	X	21.9	4	X	11.8	1
Sum	26	10702.4	107	26	4430.7	60	27	14234.9	59	26	3383.6	46	27	4754.5	26
Sum'		961.5	74		458.6	46		700.8	33		245.8	34		173.1	17

Table 8: Pricing problem heuristics used with exact pricing algorithm SP1.

6.3 Label elimination

In Section 3.2.3 methods for eliminating labels were discussed. The basic idea is to select a subset D' of deliveries of $\mathcal{O}(L)$ and test if it is possible to find a time window feasible tour from $\eta(L)$ through all the nodes in D' and ending in $2n+1$. As explained in Section 3.2.3 we use subsets D' of cardinality 1,2 and 3. The resulting label elimination rules are denoted *Label elim. 1, 2 and 3*, respectively, in the computational results below. The three elimination rules are tested on the same set of instances as considered in Section 6.2, the results are shown in Table 9. The table contains three major columns, one for each label elimination rule (the rules are applied incrementally, that is, the last column contains the results where all three elimination rules are used). The columns denoted *time* reports the time in seconds needed to solve the problem to optimality and the columns *speedup* reports the speedup relative to only using label eliminate rule 1. Blank entries in the time column indicate that the algorithm did not finish within the time limit. The tests show that the label elimination rules certainly are worthwhile, especially for hard instances. Furthermore, they seem to come "for free" - the running time did not increase for any of the instances as one might have feared. It is also clear that the simple extension of the label elimination rule presented in this paper (considering a single subset of deliveries of $\mathcal{O}(L)$ containing three elements) is able to speed up the pricing algorithm considerably. For hard instances the new elimination rule often gives a speedup of at least two, compared to the elimination criterion proposed by Dumas et al. (1991).

	Label elim. 1	Label elim. 1+2		Label elim. 1+2+3	
	time	time	speedup	time	speedup
lr101	0.3	0.3	1.0	0.3	1.0
lr102	2.8	1.5	1.8	1.5	1.9
lr103	65.9	8.1	8.2	5.9	11.1
lr104		2584.1		887.9	
lr105	0.5	0.4	1.0	0.4	1.0
lr106	1.4	1.3	1.1	1.3	1.1
lr107	2521.0	124.4	20.3	49.2	51.3
lr108		1390.4		448.2	0.0
lr109	2.0	1.8	1.1	1.8	1.1
lr110	75.1	22.3	3.4	17.1	4.4
lr111	156.7	26.4	5.9	17.1	9.2
lr112		1378.0	0.0	749.6	0.0
lc101	0.9	0.9	1.0	0.6	1.5
lc102	2064.8	604.2	3.4	339.2	6.1
lc103					
lc104					
lc105	1.5	1.3	1.1	1.3	1.1
lc106	53.3	11.8	4.5	9.3	5.7
lc107	63.4	8.4	7.6	5.4	11.7
lc108	2329.5	378.1	6.2	266.0	8.8
lc109		3882.0		2481.5	
lrc101	0.9	0.9	1.0	0.9	1.0
lrc102	1.9	1.6	1.2	1.5	1.2
lrc103	6.6	3.6	1.8	3.4	1.9
lrc104	2157.9	116.2	18.6	47.6	45.3
lrc105	1.0	1.0	1.0	1.0	1.0
lrc106	1.3	1.3	1.0	1.3	1.0
lrc107	5.0	3.4	1.5	3.3	1.5
lrc108	40.7	14.0	2.9	12.5	3.3
Sum	9554.1	10567.6		5354.9	
Average			4.0		6.9

Table 9: Label elimination rules.

6.4 Branching rule

In Section 5.3 three different branching strategies were presented. In this section the effect of these strategies is tested. Four configurations are tested, these are described in Table 10. In

Strategy name	Description
B1	branch on edges
B2	branch on vehicles + edges
B3	branch on outflow
B4	branch on vehicles + outflow

Table 10: Branching strategies

strategy *B2* and *B4* the algorithm first tries to branch on vehicles and if this is not possible then the algorithm selects the alternative branching rule.

The branching rules were tested on 14 instances where the LPM was known to yield a fractional solution in the root node. Instances from all three problem classes were tested. The results are shown in Table 11. The first column shows the name of the instance, the next five show information about the instance: *n* — number of requests, *Alg.* — relaxation/pricing problem used, *RLB* — lower bound in root node, *UB* — upper bound (optimal solution), *RLB/UB* — the ratio between lower and upper bound. For each branching strategy three columns are shown: *Opt* — indicates if the optimal solution was reached within the time limit, *#nodes* — the number of nodes explored (if the algorithm timed out, this column shows the number of nodes explored within the time limit), *time* — the total running time in seconds.

The results show that branching on vehicles is useful as all problems can be solved when using this rule. For instance B75, which is one of the PDPTW instances with a fixed cost of 10000 per vehicle, the branching rule that branch on edges does not do very well, while the branching rule that branch on vehicles solves the instance quite quickly. This is easy to explain. The fractional solution uses a fractional number of vehicles, less than 9. When branching on vehicles, two branches are created, one where at most 8 vehicles can be used and one where at lest 9 vehicles must be used. The first branch is discarded as infeasible and the other branch improves the lower bound significantly.

Altogether B4 comes out as the best strategy and it will be used in the following sections.

						B1			B2			B3			B4		
name	n	Alg.	RLB	UB	RLB/UB	Opt	#nodes	time	Opt	#nodes	time	Opt	#nodes	time	Opt	#nodes	time
lrc101	53	SP3	1697.8	1703.2	99.684%	X	7	67.9	X	7	70.8	X	9	74.0	X	7	69.6
lrc108	52	SP4	1141.1	1147.4	99.446%	X	37	1335.0	X	35	1410.1	X	43	1422.5	X	23	1117.2
LR1_2_5	106	SP3	4217.6	4221.6	99.904%	X	7	2290.1	X	7	2176.6	X	17	2967.3	X	17	2917.4
LC1_2_8	105	SP4	2682.7	2689.8	99.734%		77		X	3	898.2		116		X	3	896.6
LRC1_2_1	106	SP3	3593.3	3606.1	99.647%	X	21	4909.6	X	5	2167.5	X	19	5211.3	X	7	2323.7
LR1_4_5	206	SP1	9509.6	9517	99.922%	X	39	3678.0	X	21	1964.7	X	5	489.8	X	11	1167.5
B65	65	SP1	82615.2	82618	99.997%	X	9	63.7	X	9	63.7	X	23	126.9	X	23	126.8
D40	40	SP1	61515.8	61528	99.981%	X	37	55.9	X	37	55.9	X	35	63.5	X	35	63.1
B75	75	SP2	87514.1	92473	94.638%		1732		X	31	576.9		180		X	33	572.8
D35	35	SP2	71305.3	71308	99.996%	X	15	19.4	X	15	19.4	X	9	14.9	X	9	14.9
a5-50	50	SP2	680.8	686.62	99.149%		279		X	187	5743.6	X	129	5517.6	X	103	5016.2
a6-60	60	SP1	819.1	819.24	99.981%	X	3	584.1	X	3	586.4	X	3	560.6	X	3	557.6
b5-60	60	SP1	898.3	902.03	99.583%	X	75	3167.9	X	75	3177.3	X	23	1367.7	X	23	1360.3
b8-64	64	SP2	836.6	839.88	99.610%	X	63	1318.8	X	63	1322.0	X	25	808.0	X	25	807.0
					Sum	11	2401	17490.3	14	498	20233.1	12	636	18624.0	14	322	17010.7

Table 11: Effect of branching rules.

6.5 Cuts

This section investigates the effect of adding valid inequalities to the PDPTW relaxations SP1 – SP4. A test set consisting of 12 instances, 4 from each problem class was chosen for the experiments. Tables 12 – 15 show the quality of the lower bounds obtained using cuts and the different relaxations. The left most column gives the name of the instance, the next 8 columns show $\frac{LB}{UB} * 100$ where LB is the root lower bound and UB is the upper bound. The columns indicate the classes of cuts added: *No cuts* — no cuts added, *IPC* — infeasible paths, *CC* — capacity constraints, *FC* — fork constraints, *RC* — reachability constraints, *PC* — precedence constraints, *SPC* — strengthened precedence constraints. The column *U.Bound* shows the upper bound on the solution cost. The row *Avg.* reports the average lower bound quality and the row *Tot #cuts* shows the total number of cuts added in for all the 12 instances.

Looking at Table 12 that uses relaxation SP1 it can be seen that the lower bound obtained from this relaxation has a very high quality and adding cuts does not have a big impact. The 8 first instances are pure PDPTW instances while the 4 last are DARP instances. The cuts have the biggest impact on the DARP instances as the cuts enforce the ride time constraint that is not handled by the pricing problem. For the PDPTW instances it is only the infeasible path and the capacity cuts that have an effect. It is the strengthened infeasible paths of the type shown in equation (38) that are able to improve the lower bound. It is hard to see that the rounded capacity inequalities have an effect on the lower bound, but they do raise the lower bound on instance B30 from 51193.11 to 51193.95, and it was observed to have an impact on other instances from this class as well (but not the ones in this test). The fork, reachability, precedence and strengthened precedence inequalities did not improve the lower bound when solving pure PDPTW instances, and we have never seen these inequalities impact the lower bound when using the SP1 relaxation to solve other PDPTW instances. It seems like they all are implied by the relaxation, but we have not made any attempts to prove or disprove this. It is worth noting that Lysgaard (2005) proved that the reachability cuts for the VRPTW are redundant for a VRPTW set-partitioning relaxation based on the ESPPTWC. The FC, RC, PC and SPC inequalities have been disabled in the rest of the testing on the SP1 relaxation in the subsequent sections.

Table 13 shows that the SP2 relaxation is very close to the SP1 relaxation. For the SP2 relaxation all of the inequalities IPC, CC, FC and RC were found to have an impact when solving pure PDPTW problems while PC and SPC inequalities did not have an effect, and we have never observed these inequalities to have an effect on other pure PDPTW instances either. It seems like the PC and SPC inequalities are implied by the SP2 relaxation when solving pure PDPTW problems.

Tables 14 and 15 show that the lower bounds obtained by using the plain SP3 and SP4 relaxations are much worse than the SP1 and SP2 relaxations. It is also clear that the valid inequalities have a large impact on these relaxations as they can bring the lower bounds close to the ones for the SP1 and SP2. One notices that SP4 really is better than SP3 as predicted in Section 3.7. This is especially visible in the results where no cuts are applied. The tables also show that the strengthened precedence inequality introduced in this paper is a worthwhile contribution even though it does not seem to improve the SP1 and SP2 relaxation. Looking at Table 14 we see that SPC lower bound dominates the FC lower bound in 5 out of 12 cases and the RC lower bound in 8 out of 12 cases. The SPC inequality

	No cuts	IPC	CC	FC	RC	PC	SPC	Full	U.Bound
LR1_2_1	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	4819.1
LR1_2_9	99.34	99.36	99.34	99.34	99.34	99.34	99.34	99.36	3953.5
LC1_2_8	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	2689.8
LRC1_2_5	99.84	99.86	99.84	99.84	99.84	99.84	99.84	99.86	3715.9
A60	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92367.4
B30	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	51194.0
C30	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	51145.5
D30	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	61040.4
a3-36	98.11	98.77	98.11	99.29	98.89	98.11	98.89	99.29	583.2
a5-40	99.68	99.79	99.68	100.00	99.68	99.68	99.68	100.00	498.4
b3-36	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	603.8
b6-48	99.86	99.99	99.86	100.00	99.86	99.86	99.86	100.00	714.8
Avg.	99.74	99.81	99.74	99.87	99.80	99.74	99.80	99.88	
Tot #cuts	0	13	4	50	51	0	6	62	

Table 12: Impact of valid inequalities on SP1 .

	No cuts	IPC	CC	FC	RC	PC	SPC	Full	U.Bound
LR1_2_1	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	4819.1
LR1_2_9	99.34	99.36	99.34	99.34	99.34	99.34	99.34	99.36	3953.5
LC1_2_8	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	2689.8
LRC1_2_5	99.61	99.66	99.61	99.62	99.63	99.61	99.61	99.68	3715.9
A60	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92367.4
B30	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	51194.0
C30	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	51145.5
D30	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	61040.4
a3-36	98.11	98.77	98.11	99.29	98.89	98.11	98.89	99.29	583.2
a5-40	99.68	99.79	99.68	100.00	99.68	99.68	99.68	100.00	498.4
b3-36	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	603.8
b6-48	99.86	99.99	99.86	100.00	99.86	99.86	99.86	100.00	714.8
Avg.	99.72	99.80	99.72	99.85	99.78	99.72	99.78	99.86	
Tot #cuts	0	13	4	59	163	0	6	172	

Table 13: Impact of valid inequalities on SP2 .

would most likely improve the branch-and-cut algorithms presented by Cordeau (2005) and Ropke et al. (2005).

Tables 16 and 17 show the time used on separating inequalities in order to calculate the lower bound with the 6 different valid inequalities. Table 16 shows the time used separating inequalities when using relaxation SP1. Table 17 shows the time used separating inequalities when using relaxation SP3. It is clear that more time is spent when using SP3 as more inequalities can be added and we are going through more cut separation iterations. Compared to the overall time spend on proving the lower bound (see below) the algorithm do not spend a great deal of time on separating inequalities. The tables show that the reachability inequalities have the most time consuming separation procedure.

Tables 18 and 19 show the total time needed to prove the lower bound for the SP1 and SP3 relaxation respectively (dominance criterion (DOM1†) is used when solving the SP1 pricing problem). It is clear that the SP3 lower bound is more time consuming to obtain for these instances (except for LC1_2_8). The reason for the high running times for SP3 will be explained in Section 6.6.2.

6.6 Comparison of relaxations

This section compares the four set-partitioning relaxations to each other as well as to the branch-and-cut algorithm proposed by Ropke et al. (2005). We also test the limits of the algorithms - how large instances can be solved, how many instances from each problem class can be solved?

	No cuts	IPC	CC	FC	RC	PC	SPC	Full	U.Bound
LR1_2_1	98.58	100.00	98.58	100.00	100.00	100.00	100.00	100.00	4819.1
LR1_2_9	91.75	92.79	91.75	95.78	96.45	94.84	96.85	97.71	3953.5
LC1_2_8	98.92	99.04	98.92	99.46	99.21	99.43	99.70	99.74	2689.8
LRC1_2_5	85.51	86.77	85.51	91.65	96.31	92.07	96.66	97.27	3715.9
A60	92.53	92.58	96.20	100.00	99.91	92.66	96.45	100.00	92367.4
B30	99.67	99.76	99.79	99.99	99.97	99.81	99.99	99.99	51194.0
C30	99.90	99.93	99.99	100.00	99.97	99.91	100.00	100.00	51145.5
D30	66.95	67.00	83.46	89.10	99.93	67.00	90.51	100.00	61040.4
a3-36	92.38	95.27	92.49	99.29	98.23	96.02	98.03	99.29	583.2
a5-40	80.23	87.99	80.27	100.00	98.85	98.26	99.62	100.00	498.4
b3-36	93.82	97.85	97.98	99.97	98.61	99.01	100.00	100.00	603.8
b6-48	92.77	95.53	95.54	100.00	98.99	97.04	99.73	100.00	714.8
Avg.	91.08	92.88	93.37	97.94	98.87	94.67	98.13	99.50	
Tot #cuts	0	741	2243	3347	10226	3131	1689	5362	

Table 14: Impact of valid inequalities on SP3 .

	No cuts	IPC	CC	FC	RC	PC	SPC	Full	U.Bound
LR1_2_1	99.60	100.00	99.60	100.00	100.00	100.00	100.00	100.00	4819.1
LR1_2_9	93.38	94.13	93.38	96.40	96.92	95.89	97.39	98.06	3953.5
LC1_2_8	98.92	99.04	98.92	99.45	99.21	99.43	99.70	99.74	2689.8
LRC1_2_5	92.05	92.45	92.05	95.70	96.82	95.63	97.42	97.63	3715.9
A60	92.53	92.58	96.20	100.00	99.91	92.65	96.45	100.00	92367.4
B30	99.67	99.76	99.79	99.99	99.97	99.81	99.98	99.99	51194.0
C30	99.90	99.93	99.99	100.00	99.97	99.91	100.00	100.00	51145.5
D30	66.95	67.00	83.46	89.10	99.93	67.00	90.51	100.00	61040.4
a3-36	94.41	97.03	94.47	99.29	98.23	97.73	98.89	99.29	583.2
a5-40	81.00	88.74	81.28	100.00	98.85	98.38	99.62	100.00	498.4
b3-36	93.82	97.85	98.19	99.97	98.61	99.01	100.00	100.00	603.8
b6-48	92.77	95.56	95.55	100.00	98.99	97.02	99.73	100.00	714.8
Avg.	92.08	93.67	94.41	98.32	98.95	95.21	98.31	99.56	
Tot #cuts	0	616	2423	2892	8079	562	1268	4349	

Table 15: Impact of valid inequalities on SP4 .

	No cuts	IPC	CC	FC	RC	PC	SPC	Full
LR1_2_1	0.0	1.7	0.5	0.0	78.9	0.0	0.1	80.7
LR1_2_9	0.0	3.7	0.8	0.0	31.7	0.1	0.1	36.7
LC1_2_8	0.0	3.3	1.0	0.0	4.8	0.1	0.1	9.5
LRC1_2_5	0.0	7.3	0.8	0.0	49.8	0.1	0.1	80.0
A60	0.0	0.2	0.2	0.0	0.8	0.0	0.0	1.1
B30	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.2
C30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
D30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
a3-36	0.0	0.2	0.1	0.1	0.1	0.0	0.0	0.3
a5-40	0.0	0.2	0.1	0.0	0.0	0.0	0.0	0.3
b3-36	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.2
b6-48	0.0	0.2	0.1	0.0	0.1	0.0	0.0	0.4
Avg.	0.0	1.4	0.3	0.0	13.9	0.0	0.0	17.5

Table 16: Time spent separating valid inequalities using relaxation SP1 .

	No cuts	IPC	CC	FC	RC	PC	SPC	Full
LR1_2_1	0.0	13.6	0.5	1.6	83.3	0.4	0.2	82.3
LR1_2_9	0.0	12.0	0.4	10.9	237.3	4.8	2.2	402.1
LC1_2_8	0.0	9.1	0.4	4.1	21.2	0.5	0.5	14.1
LRC1_2_5	0.0	12.5	0.6	16.5	584.0	3.9	3.4	549.5
A60	0.0	1.9	0.8	1.4	11.2	0.5	0.3	2.2
B30	0.0	0.3	0.2	0.3	0.8	0.0	0.0	0.7
C30	0.0	0.1	0.3	0.1	0.2	0.0	0.0	0.2
D30	0.0	0.2	0.3	0.6	1.4	0.1	0.1	1.1
a3-36	0.0	0.4	0.1	0.2	0.1	0.0	0.0	0.4
a5-40	0.0	0.5	0.3	0.5	0.2	0.1	0.0	0.6
b3-36	0.0	0.3	0.4	0.2	0.1	0.1	0.0	0.3
b6-48	0.0	1.0	0.7	0.7	1.4	0.2	0.1	0.9
Avg.	0.0	4.3	0.4	3.1	78.4	0.9	0.6	87.9

Table 17: Time spent separating valid inequalities using relaxation SP3 .

	No cuts	IPC	CC	FC	RC	PC	SPC	Full
LR1_2_1	1.9	3.6	2.4	2.0	110.8	2.0	32.3	112.9
LR1_2_9	127.9	69.2	129.1	130.6	192.2	113.2	173.2	149.9
LC1_2_8	1802.1	1832.1	1838.4	1815.3	1885.4	1801.6	1888.2	1905.9
LRC1_2_5	612.0	814.7	614.1	607.6	713.3	762.9	661.0	937.0
A60	1.9	2.0	2.0	1.9	13.0	1.9	12.3	13.4
B30	0.2	0.2	0.3	0.2	1.5	0.2	1.5	1.7
C30	0.2	0.2	0.2	0.2	2.1	0.2	2.1	2.2
D30	0.2	0.3	0.3	0.3	2.4	0.3	2.4	2.5
a3-36	2.2	3.8	2.3	4.5	6.6	2.2	6.3	7.5
a5-40	2.2	2.7	2.2	2.6	5.2	2.2	5.2	5.8
b3-36	1.9	1.9	1.9	1.9	3.7	1.8	3.7	4.0
b6-48	1.6	1.9	1.7	1.7	6.5	1.6	6.5	7.1
Avg.	212.9	227.7	216.2	214.0	245.2	224.2	232.9	262.5

Table 18: Time for obtaining lower bound using valid inequalities and SP1 .

	No cuts	IPC	CC	FC	RC	PC	SPC	Full
LR1_2_1	228.6	344.2	228.8	298.8	443.0	373.5	349.4	411.2
LR1_2_9	652.0	1297.3	669.9	2826.3	9152.7	4495.2	4457.3	9021.7
LC1_2_8	349.9	490.1	349.6	511.4	814.8	647.4	687.7	663.6
LRC1_2_5	569.4	1101.2	567.8	4250.8	14229.2	5479.1	9255.7	16244.8
A60	568.0	705.5	942.3	2948.0	4667.7	2822.6	9947.1	2447.6
B30	10.9	22.2	39.8	216.8	220.1	55.8	148.2	239.8
C30	2.4	3.6	20.4	22.2	9.0	3.7	13.1	14.0
D30	23.7	59.3	329.9	1549.5	1834.8	81.9	2683.7	6560.0
a3-36	21.6	83.2	23.0	218.7	137.0	69.8	122.8	223.5
a5-40	32.1	98.1	34.2	1192.2	508.7	1454.8	1497.3	1198.1
b3-36	46.9	105.1	119.5	159.2	113.5	162.1	122.9	136.3
b6-48	101.7	187.7	370.8	690.6	580.4	661.3	507.4	602.0
Avg.	217.3	374.8	308.0	1240.4	2725.9	1358.9	2482.7	3146.9

Table 19: Time for obtaining lower bound using valid inequalities and SP3 .

6.6.1 PDPTW results

The first problem class we consider is the instances proposed by Li and Lim (2001), produced from the Solomon instances for the VRPTW. In their paper, instances with approximately 50 requests were presented and larger instances were made available on the Internet. For each instance size the set is divided into two classes: series 1 and series 2. Time windows and capacities in series 2 are constructed such that much longer routes are possible compared to series 1. When solving these instances we solely minimize distance, there are no fixed cost on using vehicles and no limit on the number of vehicles available.

Table 20 shows results for series 1 of the 50 request instances. The two first columns show the instance name and the upper bound for the instance. The remaining columns show *Opt* — if the problem was solved to optimality within the time limit, *RLB* — lower bound in root node after adding cuts, *time* — total amount of time used (in seconds), *LB* — if the bound was proved (this is interesting as the pricing problem often is so hard that getting a lower bound is very time consuming), *nodes* — the number of nodes in the branch and bound tree, *Cuts* — the number of cuts added. Notice that results for two algorithms for each of SP1 and SP2 are shown in the table (see Section 6.1). For SP1* and SP2* we only solve the root node and do not add cuts, so these algorithms only solve the problem to optimality if the LPM happens to return an integer solution, which does occur quite often.

The row *Sum* sums the number of times that optimally was reached and the number of times a lower bound was established. For the branch-and-cut algorithm a lower bound is always proved. The row *Avg.* averages the total time used (only for the instances solved to optimality) and the average number of cuts added. The entries in the *RLB* column show the root lower bound quality relative to the upper bound. Blank entries in any column indicate that the problem was not solved within the time limit.

Several comments can be made for this table. One will first notice that SP1* is much faster than SP1 and the same comment is true for SP2* vs. SP2. Therefore it would definitely be profitable to develop a branching strategy compatible with the dominance criteria within these algorithms, or to make the algorithms compatible with adding valid inequalities. Secondly it can be seen that the lower bounds obtained by relaxation SP1 and SP2 are very similar and perhaps even more surprisingly that the pricing problems for the two relaxations seem comparably hard to solve, judging from the time used. For these instances the SP3 and SP4 relaxations are slightly inferior to the SP1 and SP2 relaxations, although the difference is rather small. SP4 appears better than SP3, but the difference in lower bound is very small after cuts have been added. Looking at the cuts added for the weakest set-partitioning formulation, SP3, one sees that instances lr102, lr106, lc101 and lc106 are very easy as no cuts have to be added. This means that the precedence and pairing constraints are not binding in these instances or they are handled by the time window tightening presented in Section 5.2.

The branch-and-cut algorithm is clearly inferior to all the set-partitioning approaches for these instances as only 18 instances were solved to optimality.

All instances in the set was solved to optimality by at least one approach so no unsolved instances in this set remain.

Table 21 shows the results on the series 2 instances, and here we see a completely different picture. The most striking observation is that the cut-compatible pricing algorithms for SP1 and SP2 are not even able to prove a lower bound for any of the instances — the pricing problems are too hard. The algorithms using the stronger dominance criterion do a little better as 7 instances are solved to optimality and a tight lower bound was proven for one more instance. The SP3 and SP4 relaxations appear to be better for these instances as they are able to prove a lower bound. The overall winner for these instances, however, is the branch and cut algorithm. 17 instances remain unsolved in this data set.

Table 22 shows results for larger instances, containing around 100 requests. This set of instances also contains two series. We only show results for series 1, as we judged series 2 to be too hard. The best algorithms in this test are SP1* and SP4. Once again we see that SP1* and SP2* are looking promising and more instances could be solved to optimality with a compatible branch and bound algorithm. The branch-and-cut algorithm is performing worst in this test, at least in terms of number of instances solved to optimality, but it is not very far behind. Overall, 12 instances were solved to optimality while 18 remain unsolved.

Table 23 contains results for instances with between 200 and 500 requests. 238 of such instances exist in the dataset provided by Li and Lim (2001), but here we chose the 24 instances that we expected to be easiest to solve. We selected the instances with the tightest time windows. We did not test the branch and cut code on these instances as it is not tuned towards such large instances. We also had to turn off the reachability and strengthened precedence inequalities as the preprocessing method for calculating A_i^+, A_i^- and A_i took up a large fraction of the running time.

The results are quite encouraging as the SP1* and SP2* algorithms were able to obtain a lower bound for all instances and the SP1 and SP2 algorithms solved half of the instances to optimality. Note that two instances with 500 requests were solved to optimality. We believe that these are the largest PDPTW instances solved to optimality in the literature. The SP3 and SP4 relaxations did not do well in these tests. This is not so much because the pricing problem is hard to solve or the lower bounds are too poor, but more because the set partitioning formulations turned out to contain many columns and many cuts. We investigate this problem further for another class of problems in Section 6.6.2. 12 instances remain unsolved in this class of instances.

The last set of PDPTW instances were proposed in Ropke et al. (2005) and are similar to the ones proposed by Savelsbergh and Sol (1998). One special feature about these instances is that each vehicle has a fixed cost equal to 10000. The results for these instances are shown in Tables 24 and 25. The pricing problems for most of these instances are relatively easy, so algorithm SP1* and SP2* were not applied to these instances. Some preliminary testing showed that using pricing heuristic configuration A8 was faster than A10, so this heuristic configuration has been used for producing these results and the DARP results in Section 6.6.2. Both SP1 and SP2 produce very good results for these instances and clearly outperform the branch-and-cut algorithm - the set-partitioning formulation using SP1 and SP2 is much better at getting the number of vehicles right in the LP relaxation. The SP3 and SP4 relaxations do not do very well though. For most of the instances they fail to find a lower bound. We again refer to Section 6.6.2 for an explanation.

Even though the SP1 and SP2 relaxations do very well, they fail to solve the largest instances and the E class proves to be difficult as well. 17 out of the 75 instances are unsolved.

name	UB	Branch & Cut Opt	Branch & Cut RLB	Branch & Cut time	SP1* Opt	SP1* LB	SP1* RLB	SP1* time	SP2* Opt	SP2* LB	SP2* RLB	SP2* time	SP1 Opt	SP1 LB	SP1 RLB	SP1 time	SP1 nodes	SP1 Cuts
lr101	1650.8	X	1650.8	19.5	X	X	1650.8	0.1	X	X	1650.8	0.1	X	X	1650.8	0.3	1	0
lr102	1487.6	X	1487.6	55.7	X	X	1487.6	0.7	X	X	1487.6	0.8	X	X	1487.6	1.5	1	0
lr103	1292.7	X	1292.7	49.5	X	X	1292.7	2.2	X	X	1292.7	2.2	X	X	1292.7	5.9	1	0
lr104	1013.4	-	964.4		X	X	1013.4	7.1	X	X	1013.4	5.6	X	X	1013.4	690.9	1	0
lr105	1377.2	X	1377.1	26.6	X	X	1377.1	0.2	X	X	1377.1	0.2	X	X	1377.1	0.5	1	0
lr106	1252.6	X	1252.6	36.1	X	X	1252.6	0.8	X	X	1252.6	0.7	X	X	1252.6	1.3	1	0
lr107	1111.3	X	1108.3	151.8	X	X	1111.3	5.5	X	X	1111.3	3.3	X	X	1111.3	49.3	1	0
lr108	969.0	-	863.6		X	X	969.0	44.0	X	X	969.0	15.3	X	X	969.0	452.0	1	0
lr109	1209.0	-	1162.1		X	X	1209.0	1.3	X	X	1209.0	1.7	X	X	1209.0	1.8	1	0
lr110	1159.3	-	966.8		-	X	1157.7	2.1	-	X	1157.5	3.0	X	X	1157.7	100.1	27	0
lr111	1108.9	-	1045.7		X	X	1108.9	4.5	X	X	1108.9	3.1	X	X	1108.9	17.2	1	0
lr112	1003.8	-	739.8		X	X	1003.8	50.5	X	X	1003.8	29.2	X	X	1003.8	755.6	1	0
lc101	828.9	X	828.9	14.8	X	X	828.9	0.3	X	X	828.9	0.2	X	X	828.9	0.6	1	0
lc102	828.9	X	828.9	43.5	X	X	828.9	1.3	X	X	828.9	1.7	X	X	828.9	343.6	1	0
lc103	827.9	X	824.3	125.4	X	X	827.9	5.8	X	X	827.9	4.8	-	-				
lc104	818.6	-	709.9		X	X	818.6	943.4	X	X	818.6	177.9	-	-				
lc105	828.9	X	828.9	15.1	X	X	828.9	0.4	X	X	828.9	0.3	X	X	828.9	1.3	1	0
lc106	828.9	X	828.9	26.3	X	X	828.9	2.0	X	X	828.9	1.1	X	X	828.9	9.3	1	0
lc107	828.9	X	828.9	16.2	X	X	828.9	1.2	X	X	828.9	1.0	X	X	828.9	5.5	1	0
lc108	826.4	X	807.4	60.2	X	X	826.4	4.0	X	X	826.4	3.7	X	X	826.4	264.9	1	0
lc109	827.8	-	751.2		X	X	827.8	13.0	X	X	827.8	8.5	X	X	827.8	1907.7	1	0
lrc101	1703.2	X	1694.1	56.6	-	X	1701.9	0.3	-	X	1701.9	0.2	X	X	1701.9	0.9	3	0
lrc102	1558.1	X	1541.7	184.2	X	X	1558.1	1.1	X	X	1558.1	1.0	X	X	1558.1	1.5	1	0
lrc103	1258.7	X	1220.1	456.0	X	X	1258.7	2.2	X	X	1258.7	3.3	X	X	1258.7	3.4	1	0
lrc104	1128.4	-	998.5		X	X	1128.4	8.4	X	X	1128.4	10.1	X	X	1128.4	45.6	1	0
lrc105	1637.6	X	1632.3	61.5	X	X	1637.6	0.4	X	X	1637.6	0.7	X	X	1637.6	1.0	1	0
lrc106	1424.7	X	1369.6	1059.4	X	X	1424.7	1.0	X	X	1424.7	1.1	X	X	1424.7	1.3	1	0
lrc107	1230.1	-	1094.3		X	X	1230.1	2.5	X	X	1230.1	2.9	X	X	1230.1	3.3	1	0
lrc108	1147.4	-	928.6		X	X	1147.4	6.6	-	X	1145.4	6.5	X	X	1147.4	11.8	1	0
Sum		18			27	29			26	29			27	27				
Avg.			95.03%	136.57			99.99%	41.1			99.99%	10.8			99.99%	173.3		0.0

name	SP2 Opt	SP2 LB	SP2 RLB	SP2 time	SP2 nodes	SP2 Cuts	SP3 Opt	SP3 LB	SP3 RLB	SP3 time	SP3 nodes	SP3 Cuts	SP4 Opt	SP4 LB	SP4 RLB	SP4 time	SP4 nodes	SP4 Cuts
lr101	X	X	1650.8	10.9	1	0	X	X	1650.8	16.8	1	16	X	X	1650.8	13.6	1	0
lr102	X	X	1487.6	12.4	1	0	X	X	1487.6	29.5	1	0	X	X	1487.6	20.9	1	0
lr103	X	X	1292.7	18.0	1	0	X	X	1292.7	88.8	1	65	X	X	1292.7	36.7	1	0
lr104	X	X	1013.4	445.9	1	0	X	X	1013.4	297.5	1	301	X	X	1013.4	197.2	1	62
lr105	X	X	1377.1	7.1	1	0	X	X	1377.1	19.2	1	4	X	X	1377.1	15.9	1	4
lr106	X	X	1252.6	9.2	1	0	X	X	1252.6	20.1	1	0	X	X	1252.6	14.0	1	0
lr107	X	X	1111.3	64.9	1	0	X	X	1111.3	43.0	1	5	X	X	1111.3	35.1	1	5
lr108	X	X	969.0	389.2	1	0	X	X	969.0	169.8	1	181	X	X	969.0	181.5	1	110
lr109	X	X	1209.0	11.0	1	0	X	X	1207.9	74.4	3	207	X	X	1208.2	85.3	3	208
lr110	X	X	1157.5	192.5	45	0	-	X	1136.1		211	1335	-	X	1135.9		247	1259
lr111	X	X	1108.9	34.0	1	0	X	X	1106.7	337.4	5	367	X	X	1107.0	253.0	5	386
lr112	X	X	1003.8	518.1	1	0	-	X	974.3		34	544	-	X	975.6		45	541
lc101	X	X	828.9	7.2	1	0	X	X	828.9	24.4	1	0	X	X	828.9	13.9	1	0
lc102	X	X	828.9	480.7	1	0	X	X	828.9	56.8	1	85	X	X	828.9	75.8	1	85
lc103	-	-					X	X	827.9	116.6	1	36	X	X	827.9	99.5	1	31
lc104	-	-					-	X	816.1		13	133	-	X	816.8		13	79
lc105	X	X	828.9	8.2	1	0	X	X	828.9	39.9	1	19	X	X	828.9	39.2	1	19
lc106	X	X	828.9	15.8	1	0	X	X	828.9	34.4	1	0	X	X	828.9	55.8	1	0
lc107	X	X	828.9	13.7	1	0	X	X	828.9	80.0	1	20	X	X	828.9	113.7	1	20
lc108	X	X	826.4	378.6	1	0	X	X	826.4	41.0	1	22	X	X	826.4	77.8	1	22
lc109	X	X	827.8	2178.8	1	0	X	X	827.8	81.1	1	31	X	X	827.8	111.1	1	31
lrc101	X	X	1701.9	11.9	3	0	X	X	1698.2	68.8	7	411	X	X	1699.8	40.6	7	277
lrc102	X	X	1558.1	9.3	1	0	X	X	1558.1	45.7	1	329	X	X	1558.1	21.6	1	30
lrc103	X	X	1258.7	15.0	1	0	-	X	1258.5		3	76	X	X	1258.6	262.6	3	40
lrc104	X	X	1128.4	143.0	1	0	-	X	1127.9		5	404	X	X	1128.2	406.9	3	99
lrc105	X	X	1637.6	9.4	1	0	X	X	1637.6	34.2	1	126	X	X	1637.6	23.4	1	39
lrc106	X	X	1424.7	8.0	1	0	X	X	1424.7	64.5	1	308	X	X	1424.7	52.1	1	328
lrc107	X	X	1230.1	14.1	1	0	X	X	1230.1	93.8	1	149	X	X	1230.1	89.6	1	133
lrc108	X	X	1146.1	70.6	5	9	X	X	1141.0	1021.2	17	292	X	X	1141.2	937.5	23	282
Sum	27	27					24	29					26	29				
Avg.			99.99%	188.0		0.3			99.78%	120.8		123.9			99.79%	125.9		85.0

Table 20: Li and Lim instances, series 1. Each instance contains approximately 50 request.

		Branch & Cut			SP1*				SP2*				SP1					
name	UB	Opt	RLB	time	Opt	LB	RLB	time	Opt	LB	RLB	time	Opt	LB	RLB	time	nodes	Cuts
lr201	1253.3	X	1229.0	211.8	-	X	1253.1	147.4	-	X	1253.1	210.8	-	-				
lr202	1250.1	-	1069.9		-	-			-	-			-	-				
lr203	949.4	-	845.8		-	-			-	-			-	-				
lr204	849.1	-	689.5		-	-			-	-			-	-				
lr205	1054.1	-	982.3		-	-			-	-			-	-				
lr206	931.7	-	855.5		-	-			-	-			-	-				
lr207	903.1	-	759.7		-	-			-	-			-	-				
lr208	734.9	-	671.0		-	-			-	-			-	-				
lr209	930.6	-	845.9		-	-			-	-			-	-				
lr210	964.3	-	880.7		-	-			-	-			-	-				
lr211	884.3	-	700.9		-	-			-	-			-	-				
lc201	591.6	X	591.6	15.4	X	X	591.6	4.1	X	X	591.6	5.4	-	-				
lc202	591.6	X	591.6	17.8	X	X	591.6	160.6	X	X	591.6	1206.7	-	-				
lc203	591.2	X	591.2	46.8	-	-			-	-			-	-				
lc204	590.6	X	585.5	162.4	-	-			-	-			-	-				
lc205	588.9	X	588.9	20.7	X	X	588.9	272.6	X	X	588.9	121.4	-	-				
lc206	588.5	X	588.5	20.4	X	X	588.5	479.1	-	-			-	-				
lc207	588.3	X	588.3	19.4	X	X	588.3	392.4	X	X	588.3	822.6	-	-				
lc208	588.3	X	588.3	20.8	X	X	588.3	503.4	X	X	588.3	843.3	-	-				
lrc201	1406.9	X	1366.3	3646.8	X	X	1406.9	272.4	X	X	1406.9	473.2	-	-				
lrc202	1390.6	-	1206.3		-	-			-	-			-	-				
lrc203	1090.8	-	880.9		-	-			-	-			-	-				
lrc204	818.7	-	734.7		-	-			-	-			-	-				
lrc205	1302.2	-	1180.6		-	-			-	-			-	-				
lrc206	1159.1	-	1069.5		-	-			-	-			-	-				
lrc207	1062.1	-	909.9		-	-			-	-			-	-				
lrc208	852.8	-	674.3		-	-			-	-			-	-				
Sum		10			7	8			6	7			0	0				
Avg.			91.74%	418.23			100.00%	297.8			100.00%	578.7						

	SP2						SP3						SP4					
name	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	Tot. time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts
lr201	-	-					-	X	1242.0		21	443	-	X	1241.8		14	453
lr202	-	-					-	-					-	-				
lr203	-	-					-	-					-	-				
lr204	-	-					-	-					-	-				
lr205	-	-					-	-					-	-				
lr206	-	-					-	-					-	-				
lr207	-	-					-	-					-	-				
lr208	-	-					-	-					-	-				
lr209	-	-					-	-					-	-				
lr210	-	-					-	-					-	-				
lr211	-	-					-	-					-	-				
lc201	-	-					X	X	591.6	342.2	1	0	X	X	591.6	550.9	1	0
lc202	-	-					X	X	591.6	1387.5	1	0	X	X	591.6	1966.8	1	0
lc203	-	-					X	X	591.2	1804.3	1	0	X	X	591.2	1661.0	1	0
lc204	-	-					-	-					-	-				
lc205	-	-					X	X	588.9	274.8	1	0	X	X	588.9	306.7	1	0
lc206	-	-					X	X	588.5	686.9	1	0	X	X	588.5	310.6	1	0
lc207	-	-					X	X	588.3	1545.4	1	0	X	X	588.3	941.8	1	0
lc208	-	-					X	X	588.3	518.9	1	0	X	X	588.3	458.8	1	0
lrc201	-	-					-	X	1395.1		11	511	-	X	1395.0		5	422
lrc202	-	-					-	X	1293.4		2	486	-	-				
lrc203	-	-					-	-					-	-				
lrc204	-	-					-	-					-	-				
lrc205	-	-					-	X	1264.1		2	617	-	-				
lrc206	-	-					-	X	1115.4		4	301	-	X	1116.2		2	252
lrc207	-	-					-	X	1030.7		2	301	-	-				
lrc208	-	-					-	-					-	-				
Sum	0	0					7	13					7	10				
Avg.									98.58%	937.2		0.0			99.45%	885.2		0.0

Table 21: Li and Lim instances, series 2. Each instance contains approximately 50 request.

		Branch & Cut			SP1*				SP2*				SP1						SP2						SP3						SP4					
name	UB	Opt	RLB	time	Opt	LB	RLB	time	Opt	LB	RLB	time	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts
LR1_2_1	4819.1	X	4819.1	298	X	X	4819.1	2	X	X	4819.1	2	X	X	4819.1	4	1	0	X	X	4819.1	111	1	0	X	X	4819.1	407	1	91	X	X	4819.1	294	1	5
LR1_2_2	4093.0	-	4067.4		X	X	4093.0	81	X	X	4093.0	78	-	-					-	-					X	X	4093.0	3147	1	564	X	X	4093.0	1107	1	19
LR1_2_3	3486.9	-	3312.0		-	-			-	-			-	-					-	-					-	-					-	-				
LR1_2_4	2830.7	-	2452.8		-	-			-	-			-	-					-	-					-	-					-	-				
LR1_2_5	4221.6	X	4215.0	1123	-	X	4220.7	3	-	X	4220.7	4	X	X	4221.6	11	1	1	X	X	4221.6	129	1	1	X	X	4218.7	2888	17	602	X	X	4219.8	1292	11	280
LR1_2_6	3763.0	-	3482.1		-	-			-	X	3762.4	5766	-	-					-	-					-	-					-	-				
LR1_2_7	3112.9	-	2773.2		-	-			-	-			-	-					-	-					-	-					-	-				
LR1_2_8	2650.0	-	2149.3		-	-			-	-			-	-					-	-					-	-					-	-				
LR1_2_9	3953.5	-	3815.1		-	X	3927.5	16	-	X	3927.5	19	-	X	3928.1		101	4	-	X	3928.1		92	109	-	-					-	X	3876.8		7	1110
LR1_2_10	3391.6	-	2879.5		-	-			-	-			-	-					-	-					-	-					-	-				
LC1_2_1	2704.6	X	2704.6	179	X	X	2704.6	2	X	X	2704.6	1	X	X	2704.6	4	1	0	X	X	2704.6	114	1	0	X	X	2704.6	272	1	9	X	X	2704.6	255	1	0
LC1_2_2	2764.5	X	2753.8	5030	X	X	2764.5	18	X	X	2764.5	12	-	-					-	-					X	X	2763.5	1488	3	354	X	X	2763.7	1219	3	59
LC1_2_3	2772.2	-	2561.2		X	X	2772.2	1232	X	X	2772.2	308	-	-					-	-					-	-					-	-				
LC1_2_4	2661.4	-	1944.6		-	-			-	-			-	-					-	-					-	-					-	-				
LC1_2_5	2702.0	X	2702.0	226	X	X	2702.0	3	X	X	2702.0	2	X	X	2702.0	6	1	0	X	X	2702.0	82	1	0	X	X	2702.0	428	1	15	X	X	2702.0	437	1	18
LC1_2_6	2701.0	X	2701.0	596	X	X	2701.0	7	X	X	2701.0	5	X	X	2701.0	32	1	0	X	X	2701.0	96	1	0	X	X	2701.0	496	1	17	X	X	2701.0	505	1	11
LC1_2_7	2701.0	X	2701.0	433	X	X	2701.0	5	X	X	2701.0	4	X	X	2701.0	52	1	0	X	X	2701.0	121	1	0	X	X	2701.0	402	1	12	X	X	2701.0	516	1	12
LC1_2_8	2689.8	-	2316.8		X	X	2689.8	32	X	X	2689.8	27	X	X	2689.8	1880	1	0	X	X	2689.8	1878	1	0	X	X	2682.9	876	3	182	X	X	2682.8	894	3	179
LC1_2_9	2724.3	-	1966.8		-	X	2714.2	201	-	X	2711.8	95	-	-					-	-					-	X	2707.1		23	664	-	X	2707.2		12	656
LC1_2_10	2741.6	-	1493.7		-	X	2734.1	957	-	X	2730.7	936	-	-					-	-					-	X	2670.3		2	836	-	X	2671.6		2	850
LRC1_2_1	3606.1	X	3569.1	2535	-	X	3603.2	6	-	X	3603.2	9	X	X	3603.6	25	3	1	X	X	3603.6	221	3	1	X	X	3593.3	2316	7	607	X	X	3594.2	938	5	164
LRC1_2_2	3292.5	-	3026.6		-	X	3264.2	637	-	X	3264.2	1887	-	-					-	-					-	-					-	X	3221.1		4	1982
LRC1_2_3	3079.6	-	2529.8		-	-			-	-			-	-					-	-					-	-					-	-				
LRC1_2_4	2535.8	-	2103.1		-	-			-	-			-	-					-	-					-	-					-	-				
LRC1_2_5	3715.8	-	3333.6		-	X	3709.9	96	-	X	3701.5	65	-	X	3710.7		17	2	-	X	3703.8		13	218	-	-					-	X	3627.7		7	1379
LRC1_2_6	3360.8	-	3072.7		X	X	3360.8	25	-	X	3360.7	26	X	X	3360.8	38	1	0	X	X	3360.7	195	3	0	-	X	3265.9		4	1044	-	X	3280.1		9	1346
LRC1_2_7	3317.8	-	2720.4		-	X	3294.9	745	-	X	3280.7	453	-	X	3294.9		9	0	-	X	3286.4		5	254	-	-					-	-				
LRC1_2_8	3097.0	-	2441.6		-	X	3024.3	4533	-	-			-	-					-	-					-	-					-	-				
LRC1_2_9	3058.6	-	2261.3		-	-			-	-			-	-					-	-					-	-					-	-				
LRC1_2_10	2837.5	-	2074.2		-	-			-	-			-	-					-	-					-	-					-	-				
	Sum	8			10	19			9	19			9	12					9	12					10	13					10	16				
	Avg.		88.26%	1303			99.71%	141			99.79%	49			99.87%	228		0.2			99.83%	327		0.2			99.48%	1272		245.3			99.20%	746		74.7

Table 22: Li & Lim instances. Each instance contains approximately 100 requests, series 1.

			SP1*				SP2*				SP1					
name	n	UB	Opt	LB	RLB	time	Opt	LB	RLB	time	Opt	LB	RLB	time	nodes	Cuts
LR1_4_1	208	10639.7	X	X	10639.7	12	X	X	10639.7	9	X	X	10639.7	28	1	0
LR1_4_5	206	9517.0	-	X	9509.6	27	-	X	9509.6	19	X	X	9509.6	1360	11	1
LC1_4_1	211	7152.0	X	X	7152.0	7	X	X	7152.0	8	X	X	7152.0	25	1	0
LC1_4_5	211	7150.0	X	X	7150.0	17	X	X	7150.0	14	X	X	7150.0	45	1	0
LRC1_4_1	208	8944.6	-	X	8848.4	20	-	X	8848.4	19	-	X	8848.5		356	8
LRC1_4_5	207	8667.9	-	X	8485.5	202	-	X	8472.2	143	-	X	8485.9		26	4
LR1_6_1	317	22515.4	X	X	22515.4	52	X	X	22515.4	41	X	X	22515.4	120	1	0
LR1_6_5	313	20439.9	-	X	20385.6	231	-	X	20385.6	217	-	X	20386.5		9	3
LC1_6_1	315	14095.6	X	X	14095.6	19	X	X	14095.6	19	X	X	14095.6	82	1	0
LC1_6_5	314	14086.3	X	X	14086.3	35	X	X	14086.3	27	X	X	14086.3	113	1	0
LRC1_6_1	313	17789.2	-	X	17659.6	113	-	X	17652.5	109	-	X	17659.6		69	10
LRC1_6_5	315	16645.8	-	X	16257.1	1063	-	X	16237.7	662	-	X	16258.3		2	2
LR181	421	39291.3	X	X	39291.3	149	X	X	39291.3	143	X	X	39291.3	498	1	0
LR185	419	34656.0	-	X	34480.2	432	-	X	34480.2	352	-	-				
LC181	420	25184.4	X	X	25184.4	44	X	X	25184.4	37	X	X	25184.4	188	1	0
LC185	421	25211.2	X	X	25211.2	64	X	X	25211.2	50	X	X	25211.2	220	1	0
LRC181	411	31836.5	-	X	31673.0	218	-	X	31672.3	220	-	X	31679.2		20	6
LRC185	418	30814.1	-	X	29993.1	3758	-	X	29948.5	3756	-	-				
LR1101	527	56806.0	-	X	56740.8	234	-	X	56740.8	233	-	X	56740.8		11	0
LR1105	524	52944.2	-	X	52310.9	1346	-	X	52310.2	1231	-	-				
LC1101	527	42488.6	X	X	42488.6	74	X	X	42488.6	78	X	X	42488.6	341	1	0
LC1105	529	42477.4	X	X	42477.4	110	X	X	42477.4	100	X	X	42477.4	411	1	0
LRC1101	527	48551.1	-	X	47982.7	353	-	X	47973.1	424	-	X	47990.6		9	11
LRC1105	526	49392.6	-	X	48224.8	5056	-	X	48131.4	5621	-	-				
		Sum	11	24			11	24			12	20				
		Avg.			99.4%	52.8			99.3%	47.9			99.6%	286.2		0.1

	SP2						SP3						SP4					
name	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts
LR1_4_1	X	X	10639.7	26	1	0	-	-					X	X	10639.7	1318	1	31
LR1_4_5	X	X	9509.6	811	9	1	-	-					-	-				
LC1_4_1	X	X	7152.0	27	1	0	X	X	7152.0	2020	1	4	X	X	7152.0	488	1	0
LC1_4_5	X	X	7150.0	40	1	0	X	X	7150.0	2497	1	17	X	X	7150.0	1365	1	17
LRC1_4_1	-	X	8848.5		379	6	-	-					-	X	8803.0		7	740
LRC1_4_5	-	X	8474.7		40	19	-	-					-	-				
LR1_6_1	X	X	22515.4	108	1	0	-	-					X	X	22515.4	6764	1	40
LR1_6_5	-	X	20386.5		10	3	-	-					-	-				0
LC1_6_1	X	X	14095.6	83	1	0	-	-					X	X	14095.6	2123	1	0
LC1_6_5	X	X	14086.3	100	1	0	-	-					X	X	14086.3	4419	1	21
LRC1_6_1	-	X	17652.5		81	14	-	-					-	-				
LRC1_6_5	-	X	16240.0		2	16	-	-					-	-				
LR181	X	X	39291.3	304	1	0	-	-					-	-				
LR185	-	X	34507.5		2	3	-	-					-	-				
LC181	X	X	25184.4	175	1	0	-	-					X	X	25184.4	5281	1	0
LC185	X	X	25211.2	193	1	0	-	-					-	-				
LRC181	-	X	31678.5		20	8	-	-					-	-				
LRC185	-	-					-	-					-	-				
LR1101	-	X	56740.8		11	0	-	-					-	-				
LR1105	-	X	52348.9		2	10	-	-					-	-				
LC1101	X	X	42488.6	333	1	0	-	-					-	-				
LC1105	X	X	42477.4	388	1	0	-	-					-	-				
LRC1101	-	X	47984.2		10	17	-	-					-	-				
LRC1105	-	-					-	-					-	-				
Sum	12	22					2	2					7	8				
Avg.			99.5%	215.7		0.1			100.0%	2259		10.5			99.8%	3108		15.6

Table 23: Large scale Li & Lim instances. 200-500 requests.

		Branch & Cut			SP1						SP2						SP3						SP4					
name	UB	Opt	RLB	time	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts
A30	51317.7	X	51317.7	3.1	X	X	51317.7	0.2	1	0	X	X	51317.7	1.6	1	0	X	X	51317.7	12.0	1	61	X	X	51317.7	13.9	1	61
A35	51343.9	X	51343.9	5.9	X	X	51343.9	0.5	1	0	X	X	51343.9	2.7	1	0	X	X	51343.9	406.7	1	214	X	X	51343.9	409.2	1	214
A40	61609.9	X	61609.9	8.3	X	X	61609.9	0.6	1	4	X	X	61609.9	4.2	1	4	X	X	61609.9	410.1	1	231	X	X	61609.9	397.9	1	231
A45	61693.5	-	51814.1		X	X	52716.1	5.5	3	2	X	X	52716.1	10.1	3	3	-	-					-	-				
A50	71932.6	X	71932.6	24.6	X	X	71932.6	1.4	1	0	X	X	71932.6	7.7	1	0	X	X	71932.6	5841.7	1	501	-	-				
A55	82185.9	X	82143.0	214.6	X	X	82185.9	3.2	1	15	X	X	82185.9	13.4	1	17	-	-					-	-				
A60	92367.4	X	92367.4	41.4	X	X	92367.4	2.2	1	0	X	X	92367.4	13.7	1	0	X	X	92367.4	2498.5	1	270	X	X	92367.4	2965.1	1	270
A65	82331.8	X	82331.8	77.0	X	X	82331.8	4.5	1	0	X	X	82331.8	18.7	1	0	-	-					-	-				
A70	112459.0	X	107528.1	202.7	X	X	107488.7	7.6	3	4	X	X	107488.7	27.8	3	4	-	-					-	-				
A75	92526.3	X	92494.0	990.9	X	X	92526.3	12.8	1	0	X	X	92526.3	31.8	1	0	-	-					-	-				
A100	123515.5	-	123435.2		X	X	123512.3	100.8	7	9	X	X	123512.3	219.4	9	23	-	-					-	-				
A125	134297.7	-	134214.4		X	X	134293.2	714.1	47	32	X	X	134293.2	1634.7	37	30	-	-					-	-				
A150	135062.5	-	124890.2		-	X	135056.3		222	60	X	X	135056.3	5224.6	49	39	-	-					-	-				
A175	176052.7	-	165851.1		-	X	176013.1		199	69	-	X	176013.1		19	33	-	-					-	-				
A200	206856.7	-	166033.6		-	X	206801.8		133	78	-	X	206801.8		9	11	-	-					-	-				
B30	51194.0	X	51191.1	4.8	X	X	51194.0	0.3	1	4	X	X	51194.0	1.7	1	4	X	X	51191.1	401.7	3	273	X	X	51190.6	353.2	5	241
B35	61400.4	X	56446.9	8.2	X	X	56448.2	0.6	3	0	X	X	56448.2	2.8	3	0	X	X	56448.2	1160.5	3	348	X	X	56448.2	867.6	3	360
B40	51421.8	X	44077.0	33.8	X	X	46481.8	1.9	3	3	X	X	46481.8	5.3	3	3	-	-					-	-				
B45	61787.8	X	61767.5	38.2	X	X	61780.8	4.4	11	13	X	X	61780.8	8.8	5	12	-	-					-	-				
B50	71890.3	X	71873.7	56.2	X	X	71889.1	4.3	7	13	X	X	71889.1	11.9	7	14	-	-					-	-				
B55	82081.3	X	82077.5	46.0	X	X	82081.3	2.3	1	0	X	X	82081.3	11.0	1	0	-	-					-	-				
B60	102324.5	X	102322.1	74.6	X	X	102324.5	2.0	1	0	X	X	102324.5	14.2	1	0	X	X	102324.5	6922.8	3	981	X	X	102324.5	6837.9	3	1001
B65	82618.0	X	82559.7	5748.3	X	X	82615.2	23.2	19	0	X	X	82615.2	37.1	13	0	-	-					-	-				
B70	92642.4	X	92605.4	462.9	X	X	92640.9	23.0	11	16	X	X	92640.9	37.3	5	14	-	-					-	-				
B75	92472.7	-	82626.8		X	X	87514.2	51.7	15	10	X	X	87512.9	83.4	13	9	-	-					-	-				
B100	113564.9	-	103757.2		X	X	108590.6	242.4	33	37	X	X	108589.5	259.0	9	17	-	-					-	-				
B125	134514.9	-	134369.3		X	X	134512.5	3212.5	353	32	X	X	134512.5	1108.7	25	14	-	-					-	-				
B150	144663.4	-	136401.0		X	X	141417.4	2042.2	27	32	X	X	141418.1	1818.8	9	12	-	-					-	-				
B175	165996.6	-	142472.2		-	X	161000.6		77	178	-	X	161000.7		17	64	-	-					-	-				
B200	206642.6	-	154459.4		-	X	206593.6		73	47	-	X	206593.6		7	13	-	-					-	-				
C30	51145.5	X	51145.5	3.3	X	X	51145.5	0.2	1	0	X	X	51145.5	2.2	1	0	X	X	51145.5	14.5	1	84	X	X	51145.5	15.9	1	84
C35	51236.0	X	51226.3	16.0	X	X	51235.2	0.8	3	1	X	X	51233.5	4.1	3	0	-	-					-	-				
C40	61474.3	-	51534.1		X	X	54048.2	1.7	3	0	X	X	54048.2	6.7	3	0	-	-					-	-				
C45	81406.4	X	81406.4	30.5	X	X	81406.4	1.3	1	0	X	X	81406.4	7.4	1	0	-	-					-	-				
C50	61933.6	X	61893.7	265.2	X	X	61933.2	7.2	3	4	X	X	61933.2	14.1	3	4	-	-					-	-				
C55	61931.2	X	61880.2	1274.5	X	X	61931.2	10.0	1	0	X	X	61931.2	20.2	1	0	-	-					-	-				
C60	72101.3	X	72022.7	6982.1	X	X	72101.3	6.6	1	0	X	X	72101.3	22.4	1	0	-	-					-	-				
C65	82319.7	-	82152.7		X	X	82316.5	76.8	15	3	X	X	82316.5	81.4	13	3	-	-					-	-				
C70	92612.3	-	82630.5		X	X	87683.4	34.9	5	19	X	X	87684.0	70.6	7	25	-	-					-	-				
C75	92712.5	-	92546.0		X	X	92708.7	81.7	13	8	X	X	92708.7	111.1	13	2	-	-					-	-				
C100	113373.2	-	103186.0		-	X	105477.2		795	99	-	X	105477.4		449	73	-	-					-	-				
C125	153862.8	-	143650.3		X	X	148890.1	1076.4	41	12	X	X	148890.1	1078.1	19	4	-	-					-	-				
C150	174895.8	-	174571.8		-	X	174880.8		283	60	-	X	174880.8		62	12	-	-					-	-				
C175	175876.5	-	175410.6		-	X	175794.0		103	3	-	X	175794.0		20	0	-	-					-	-				
C200	196432.4	-	185722.7		-	X	196314.7		40	40	-	X	196314.9		6	14	-	-					-	-				
	sum	24		16613.0	36	45		7761.6		273	37	45		12028.3		257	9	9		17668.3		2963	8	8		11860.6		2462
	avg.			692.2				215.6		8				325.1		7				1963.1		329				1482.6		308

Table 24: Instances proposed by Ropke, Cordeau and Laporte Ropke et al. (2005). Class A, B and C

		Branch & Cut			SP1						SP2						SP3						SP4					
name	UB	Opt	RLB	time	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts
D30	61040.4	X	61032.9	11.0	X	X	61040.4	0.3	1	0	X	X	61040.4	2.5	1	0	X	X	61040.4	6504.9	1	822	X	X	61040.4	6751.6	1	822
D35	71308.4	X	71256.1	2078.8	X	X	71305.3	3.2	9	0	X	X	71305.3	5.7	9	0	-	-					-	-				
D40	61527.5	-	61451.3		X	X	61515.8	16.1	29	9	X	X	61515.8	17.8	25	5	-	-					-	-				
D45	81602.0	X	81597.3	31.8	X	X	81600.6	3.4	11	0	X	X	81600.6	10.6	11	0	X	X	81600.5	6171.1	3	641	-	X	81600.6		5	630
D50	71761.8	X	71750.5	68.6	X	X	71761.8	4.6	1	0	X	X	71761.8	13.7	1	0	-	-					-	-				
D55	72052.5	-	71982		X	X	72052.5	10.9	1	0	X	X	72052.5	18.5	1	0	-	-					-	-				
D60	82307.1	X	82291.3	144.0	X	X	82307.1	4.0	1	0	X	X	82307.1	20.1	1	0	-	-					-	-				
D65	82201.5	-	82076.9		X	X	82201.5	19.0	1	0	X	X	82201.5	30.1	1	0	-	-					-	-				
D70	82624.2	-	82489.5		X	X	82623.9	44.7	5	0	X	X	82623.9	62.2	3	0	-	-					-	-				
D75	92971.7	-	92751.1		X	X	92971.7	42.5	1	4	X	X	92971.7	71.4	1	3	-	-					-	-				
D100	103449.2	-	103065.3		X	X	103449.1	310.9	3	2	X	X	103449.1	296.3	3	2	-	-					-	-				
D125	174237.6	-	173986.2		-	X	174205.0		696	63	-	X	174205.0		130	25	-	-					-	-				
D150	154832.8	-	70675.2		-	X	148352.3		41	33	-	X	148352.3		22	46	-	-					-	-				
D175	175708.2	-	98833.8		-	X	165802.9		4	21	-	X	165802.9		4	17	-	-					-	-				
D200	176370.0	-	140779.5		-	X	176198.5		10	14	-	X	176198.5		4	10	-	-					-	-				
E30	41258.0	X	41247.4	11.6	X	X	41258.0	0.4	1	0	X	X	41258.0	2.2	1	0	-	X	41251.2		13	513	-	X	41249.9		12	528
E35	71312.6	-	61309.2		X	X	66333.0	0.7	3	0	X	X	66333.0	4.2	3	0	-	-					-	-				
E40	61512.4	X	61461.9	1677.6	X	X	61511.9	2.8	3	2	X	X	61511.9	6.9	3	2	-	-					-	-				
E45	61472.7	X	61424.0	2842.9	X	X	61472.7	4.9	1	0	X	X	61472.7	10.4	1	0	-	-					-	-				
E50	81784.3	X	81734.9	2621.3	X	X	81780.3	48.3	45	9	X	X	81780.3	64.5	51	6	-	-					-	-				
E55	91920.7	-	91795.4		X	X	91915.8	92.7	63	21	X	X	91915.8	176.0	109	26	-	-					-	-				
E60	71998.8	-	71872.4		X	X	71990.7	904.2	245	32	X	X	71990.7	783.1	241	51	-	-					-	-				
E65	62315.7	-	62009.8		X	X	62315.0	82.3	5	8	X	X	62315.0	133.7	7	12	-	-					-	-				
E70	82590.6	-	82364.9		X	X	82577.2	241.8	13	7	X	X	82577.2	249.9	19	5	-	-					-	-				
E75	112464.8	-	102290.2		X	X	107486.6	229.0	5	33	X	X	107486.5	194.7	5	34	-	-					-	-				
E100	103367.3	-	92862.7		-	X	93621.8		9	7	-	X	93621.8		11	17	-	-					-	-				
E125	133937.7	-	133419.4		-	X	133898.1		23	21	-	X	133898.1		28	13	-	-					-	-				
E150	134704.3	-	133735.8		-	-					-	-					-	-					-	-				
E175	165800.6	-	164939.9		-	-					-	-					-	-					-	-				
E200	156202.0	-	71105.9		-	-					-	-					-	-					-	-				
	sum	9		9487.4	21	27		2066.7		127	21	27		2174.2		146	2	3		12676.1		1463	1	3		6751.6		822
	avg.			1054.2				98.4		6				103.5		7				6338.0		732				6751.6		822

Table 25: Instances proposed by Ropke, Cordeau and Laporte Ropke et al. (2005). Class D, E

6.6.2 DARP results

Tables 26 and 27 shows the results of applying the column generation algorithm to the DARP instances.

The ride time constraints are not handled in any of the pricing algorithms proposed in this paper. Instead, these must be handled by dynamically added inequalities. Using the SP1 or SP2 pricing problems one only needs to add infeasible path inequalities to obtain feasible DARP solutions, but the other inequalities can improve the quality of the lower bound as well.

The pricing problems for the DARP instances are easy to solve so the pricing algorithms SP1* and SP2* have not been applied to these problems. Inspecting the tables one first notices that many more cuts can be added to the SP1 and SP2 models compared to when solving pure PDPTW instances. The reason is obviously that the cuts ensure that ride time constraints are satisfied. One also notices that the lower bounds obtained with SP2 virtually are as good as the ones obtained with SP1. In three cases SP1 is better than SP2 (a4-48, a6-72, b8-96), but SP2 is better than SP1 in three other cases (a8-64, a8-80, 8-64). The cases where SP2 is better than SP1 occur because heuristic separation routines are used for fork and capacity inequalities.

The SP3 and SP4 formulations perform quite poorly on the DARP instances - for more than half of the instances they do not even establish a lower bound within the time limit. Table 28 contains detailed statistics for the solution of a typical instance (a6-48). This table reveals why SP3 and SP4 perform so badly. The table contains a line for each of the four lower bounds. All of the lower bounds solve the problem in the root node. The columns should be interpreted as follows *cols* - number of columns generated, *iter* - number of column generation iterations, *cuts* - number of cuts added, *time* total time to solve the root node (in seconds). The next five columns show how time is distributed among the major components of the algorithm: *preprocessing* - time spent doing preprocessing (finding A_i^- and A_i^+, performing time window reductions, etc.), *LP* - time spent in LP solver, *pricing heur* - time spent in the pricing heuristics, *pricing exact* - time spent in exact pricing algorithm, *cut generation* - time spent separating valid inequalities, *other* - time spent on bookkeeping and updating the model with new columns and rows.

It is clear that SP3 and SP4 perform many column generation iterations and generate numerous columns and consequently spend almost all of their time in the LP solver. There are several ways to improve on this. The simplest improvement would be to implement a column management routine that removes unpromising columns from the LP formulation. This is not done currently, so all generated variables are present in the LP. Savelsbergh and Sol (1998) describe one way of managing columns.

In order to reduce the number of column generation iterations, two approaches can be taken. One is to try to fine tune the choice between generating variables and cuts. It might be beneficial to generate cuts earlier compared to what is being done now. A more promising, but possibly more complicated improvement would be to use stabilized column generation (see du Merle et al. (1999)).

Compared to the branch-and-cut (BAC) algorithm presented in Ropke et al. (2005) one can see that the BAC algorithm performs better than the branch-and-cut-and-price (BCP) algorithm as all instances were solved by BAC while two are unsolved using BCP. It should be noted that the BAC algorithm was allowed to spend longer time on a8-96 than the BCP algorithm. One should also note that the BCP algorithm often obtains significantly better lower bounds than the BAC.

		Branch & Cut			SP1						SP2						SP3						SP4					
name	UB	Opt	RLB	time	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts
a2-16	294.2	X	294.2	0.6	X	X	294.2	0.3	1	13	X	X	294.2	0.3	1	13	X	X	294.2	1.5	1	23	X	X	294.2	1.7	1	23
a2-20	344.8	X	344.8	1.1	X	X	344.8	0.8	1	27	X	X	344.8	0.9	1	27	X	X	344.8	14.8	1	54	X	X	344.8	11.1	1	53
a2-24	431.1	X	430.3	2.6	X	X	430.4	2.2	3	21	X	X	430.4	3.0	3	21	X	X	430.3	200.5	7	81	X	X	430.3	164.2	5	71
a3-24	344.8	X	344.8	2.1	X	X	344.8	1.6	1	52	X	X	344.8	1.3	1	52	X	X	344.8	36.5	1	115	X	X	344.8	49.3	1	113
a3-30	494.8	X	494.8	4.7	X	X	494.8	2.6	1	24	X	X	494.8	3.0	1	24	X	X	494.8	208.7	1	133	X	X	494.8	80.1	1	114
a3-36	583.2	X	579.0	9.5	X	X	579.0	13.1	7	58	X	X	579.0	12.3	7	64	X	X	579.0	645.8	5	181	X	X	579.0	557.5	7	134
a4-32	485.5	X	485.5	5.3	X	X	485.5	3.7	1	73	X	X	485.5	3.8	1	73	X	X	485.5	789.0	1	280	X	X	485.5	378.6	1	283
a4-40	557.7	X	553.9	17.6	X	X	556.7	10.9	3	78	X	X	556.6	12.2	3	77	-	X	554.9		5	231	X	X	554.9	5863.0	5	261
a4-48	668.8	X	666.5	35.8	X	X	668.1	40.9	3	85	X	X	668.1	45.7	5	85	-	-					-	-				
a5-40	498.4	X	498.4	11.0	X	X	498.4	5.6	1	3	X	X	498.4	6.4	1	3	X	X	498.4	1274.6	1	225	X	X	498.4	740.0	1	255
a5-50	686.6	X	680.0	50.4	X	X	680.8	623.0	113	241	X	X	680.8	544.9	101	250	-	-					-	-				
a5-60	808.4	X	804.1	102.5	X	X	808.4	60.4	1	103	X	X	808.4	63.4	1	102	-	-					-	-				
a6-48	604.1	X	604.1	28.3	X	X	604.1	18.3	1	121	X	X	604.1	17.4	1	121	-	-					-	-				
a6-60	819.2	X	816.2	106.6	X	X	819.1	58.4	3	110	X	X	819.1	54.9	3	110	-	-					-	-				
a6-72	916.0	X	910.1	210.9	X	X	914.4	1646.2	33	320	X	X	913.6	1721.6	31	337	-	-					-	-				
a7-56	724.0	X	718.5	103.7	X	X	720.9	70.1	23	145	X	X	720.9	63.0	17	186	-	-					-	-				
a7-70	889.1	X	886.7	201.0	X	X	888.8	146.9	7	146	X	X	888.8	135.1	7	137	-	-					-	-				
a7-84	1033.4	X	1025.2	547.9	X	X	1028.6	2468.5	85	367	X	X	1028.6	1436.5	57	374	-	-					-	-				
a8-64	747.5	X	743.7	233.2	X	X	747.1	58.3	3	122	X	X	747.3	53.2	3	146	-	-					-	-				
a8-80	945.7	X	938.1	589.7	-	X	940.1		201	534	-	X	940.3		270	495	-	-					-	-				
a8-96	1232.6	X	1213.4	11585.4	-	X	1224.5		37	831	-	X	1224.5		31	780	-	-					-	-				
	Sum	21			19	21					19	21					8	9					9	9				
	Avg.		99.56%	659.5			99.79%	275.4		111.0			99.79%	219.9		115.9			99.84%	396.4		136.5			99.84%	871.7		145.2

Table 26: DARP instances, type A.

		Branch & Cut			SP1						SP2						SP3						SP4					
name	UB	Opt	RLB	time	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts	Opt	LB	RLB	time	nodes	Cuts
b2-16	309.4	X	308.1	0.8	X	X	309.4	0.3	1	11	X	X	309.4	0.3	1	11	X	X	309.0	26.8	9	61	X	X	309.0	15.8	5	67
b2-20	332.6	X	332.6	0.6	X	X	332.6	0.4	1	0	X	X	332.6	0.4	1	0	X	X	332.6	2.4	1	12	X	X	332.6	2.3	1	12
b2-24	444.7	X	444.5	2.0	X	X	444.5	1.5	3	5	X	X	444.5	1.7	3	5	X	X	444.5	85.8	3	63	X	X	444.5	42.1	3	47
b3-24	394.5	X	392.9	2.0	X	X	392.2	2.0	17	11	X	X	392.2	2.0	17	11	X	X	391.9	57.1	13	118	X	X	391.9	60.7	11	112
b3-30	531.4	X	531.4	2.8	X	X	531.4	1.7	1	0	X	X	531.4	1.9	1	0	X	X	531.4	67.5	1	48	X	X	531.4	50.2	1	47
b3-36	603.8	X	603.8	4.8	X	X	603.8	3.8	1	0	X	X	603.8	3.7	1	0	X	X	603.8	149.5	1	66	X	X	603.8	132.6	1	66
b4-32	494.8	X	494.8	3.1	X	X	494.8	2.0	1	0	X	X	494.8	2.1	1	0	X	X	494.8	7.8	1	50	X	X	494.8	8.9	1	50
b4-40	656.6	X	656.6	8.4	X	X	656.6	5.2	1	10	X	X	656.6	6.3	1	10	X	X	656.6	451.5	1	173	X	X	656.6	337.6	1	183
b4-48	673.8	X	671.9	26.6	X	X	673.2	24.2	3	31	X	X	673.2	23.1	3	31	-	-					-	-				
b5-40	613.7	X	611.1	18.9	X	X	613.7	4.2	1	5	X	X	613.7	4.8	1	5	-	-					X	X	613.7	6945.5	3	446
b5-50	761.4	X	756.2	43.6	X	X	761.4	12.0	1	0	X	X	761.4	11.6	1	0	-	-					-	-				
b5-60	902.0	X	893.9	102.6	X	X	898.3	178.1	29	66	X	X	898.3	192.8	33	78	-	-					-	-				
b6-48	714.8	X	714.8	13.7	X	X	714.8	6.7	1	7	X	X	714.8	7.2	1	7	X	X	714.8	598.4	1	189	X	X	714.8	483.8	1	186
b6-60	860.1	X	860.1	42.9	X	X	860.1	18.5	1	0	X	X	860.1	18.4	1	0	-	-					-	-				
b6-72	978.5	X	963.1	868.2	X	X	975.7	632.8	73	70	X	X	975.7	402.7	41	66	-	-					-	-				
b7-56	824.0	X	808.3	1103.3	X	X	822.2	70.1	27	30	X	X	822.2	66.2	39	34	-	-					-	-				
b7-70	912.6	X	907.2	165.6	X	X	911.1	73.1	15	12	X	X	911.1	65.1	9	22	-	-					-	-				
b7-84	1203.4	X	1193.2	382.2	X	X	1202.0	302.6	11	18	X	X	1202.0	237.6	9	18	-	-					-	-				
b8-64	839.9	X	834.7	158.2	X	X	836.6	123.2	65	56	X	X	836.9	105.1	57	29	-	-					-	-				
b8-80	1036.3	X	1032.6	190.5	X	X	1036.2	69.9	3	6	X	X	1036.2	73.1	3	6	-	-					-	-				
b8-96	1185.5	X	1165.1	5059.4	X	X	1182.0	3859.2	191	93	X	X	1181.5	3403.5	225	112	-	-					-	-				
	Sum	21			21	21					21	21					9	9					10	10				
	Avg.		99.49%	390.5			99.88%	256.7		20.5			99.88%	220.4		21.2			99.91%	160.7		86.7			99.92%	807.9		121.6

Table 27: DARP instances, type B.

					time distribution					
					pre-		pricing	pricing	cut	
	cols	iter	cuts	time	processing	LP	heur.	Exact	generation	other
SP1	2096	383	85	18.3	27.9%	11.2%	52.1%	0.0%	5.4%	3.4%
SP2	2060	347	85	17.4	29.1%	10.4%	51.7%	0.0%	5.5%	3.3%
SP3	80584	8613	704	27884.9	0.0%	95.4%	1.0%	0.0%	0.0%	3.6%
SP4	75078	9148	592	22696.5	0.0%	95.8%	1.8%	0.0%	0.0%	2.3%

Table 28: Statistics for DARP instance a6-48.

7 Conclusion

In the beginning of the paper we raised the question: How do the four relaxations presented in this paper compare to each other and how do they compare to a pure branch and cut approach? Can one approach be recommended as "the best"?

The computational tests bring us closer to answering these questions. Table 29 gives an overview of the number of instances solved to optimality for the different relaxations and instance classes. Each line in the table corresponds to a class of instances, thus LL50-1 and LL50-2 correspond to Tables 20 and 21. LL100 and LL200-500 correspond to Tables 22 and 23 while SaSo correspond to Tables 24 and 25 and DARP correspond to Tables 26 and 27. For the SP1 relaxations we have reported the union of solutions found by SP1 and SP1* algorithms. This seems fair as we could run the SP1* first and switch to SP1 if the solution to the LPM in the root is fractional. The same has been done for SP2. We see that SP1 and SP2 come out as winners of this test.

This leads to an interesting observation: the SP1 and SP2 lower bounds are very close and their pricing problems are roughly equally difficult with the algorithms available for solving shortest path problems. That their lower bounds were going to be close was already hinted by the fact that the cycles that can occur in the solutions to the SPPTWCPD contain at least four nodes, but there are also results in the literature that suggest that SP1 should be much stronger than SP2. Sol (1994), page 73 showed that the SP2 relaxation can be half the value of the SP1 relaxation.

At first sight it seems like the ESPPTWCPD should be much harder to solve compared to the SPPTWCPD due to the many extra resources. This does not turn out to be the case. The best explanation is probably, that although the domination check is weaker for the ESPPTWCPD than for the SPPTWCPD, the ESPPTWCPD has another advantage: the SPPTWCPD algorithm has to loop around in negative cycles - this implies that it has to store and extend sets of labels whose corresponding paths only differ by how many times a certain negative cycle has been traversed, while the ESPPTWCPD algorithm never does such looping.

The branch-and-cut algorithm is worthwhile for some problem classes, but overall SP1 and SP2 are preferable. The SP4 relaxation has the advantage in a few cases but it is not really competitive. However, it does seem like further programming efforts would be able to improve on the SP4 performance. Overall the SP4 relaxation is preferable to the SP3 relaxation. The SP4 relaxation is interesting as it often is easier to solve its pricing problem than solving the SP1 pricing problem but its resulting lower bound is of poorer quality. It is better than solving the 2-index model proposed in Ropke et al. (2005) or the 3-index model proposed in Cordeau (2005) though. It can be seen as a relaxation in between the two lower bounds.

The experiments with pricing heuristics demonstrated that it is useful to consider more

	BAC	SP1	SP2	SP3	SP4
LL50-1	18	29	29	24	26
LL50-2	10	7	6	7	7
LL100	8	12	11	10	10
LL200-500	-	12	12	2	7
SaSo	31	57	58	11	9
DARP	42	40	40	17	19
Sum	109	157	156	71	78

Table 29: Overview of number of optimal solutions

advanced pricing heuristics - significant speed ups can be obtained. More research could be done in this field though. It seems like an automatic selection of the heuristic to use to solve the pricing problem would be helpful - some pricing problems are easier than others.

There are ample opportunities for further research. The most obvious line of research is to find a branching rule compatible with the (DOM1') and (DOM2) dominance criteria or to find a way to perturb the costs d_{ij} such that the dominance criteria can be used with valid inequalities. Dumas et al. (1991) proposed a branching rule, but it creates $n_r + 2$ branches where n_r is the number of requests served in the route that is chosen for branching. Thus the branch and bound tree may grow very rapidly. A better approach might be to branch on time windows as proposed for the vehicle routing problem with time windows and backhauls by Gélinas et al. (1995).

The valid inequalities improve the SP3 and SP4 lower bounds significantly, but none of the inequalities used in this paper are able to influence the SP1 lower bound much. One candidate is the 2-path cuts for the VRPTW proposed by Kohl et al. (1999). It is possible to strengthen the cut when used for the PDPTW so it appears to be promising.

Improving the performance of the SP3 and SP4 relaxations as described in Section 6.6.2 would be interesting. It is also possible to deduce new, tighter relaxations from SP4. For example the pricing problem ESPPTWP could be constrained even more by demanding that an even number of nodes must be visited on every path or even stronger: that the number of pickup and delivery nodes must be equal on every path.

Proving or disproving theorems about the relationships between classes of valid inequalities and relaxations obtained by set-partitioning, put forward in Section 6.5, would be an interesting contribution.

We can conclude that several large-scale instances can be solved with the current approach and that many of the proposed test instances can be solved to optimality. We hope that the unsolved instances will challenge future researchers in the years to come.

References

E. Balas, M. Fischetti, and W.R. Pulleyblank. The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming*, 68:241–265, 1995.

C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance. Branch-and-price: Column generation for solving integer programs. *Operations Research*, 46: 316–329, 1998.

J. Bramel and D. Simchi-Levi. Set-covering-based algorithms for the capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 9, pages 85–108. SIAM, Philadelphia, 2002.

A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. Working Paper, ILOG, Madrid, 2003, 2003.

J.-F. Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 2005. Forthcoming.

J.-F. Cordeau, G. Laporte, J.-Y. Potvin, and M.W.P. Savelsbergh. Transportation on demand. In C. Barnhart and G. Laporte, editors, *Transportation*, Handbooks in Operations Research and Management Science. Elsevier, Amsterdam, 2005. Forthcoming.

G. Desaulniers, J. Desrosiers, A. Erdmann, M.M. Solomon, and F. Soumis. VRP with pickup and delivery. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 9, pages 225–242. SIAM, Philadelphia, 2002.

G. Desaulniers, J. Desrosiers, I. Ioachim, M.M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T.G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Kluwer, Norwell, MA, 1998.

M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992.

J. Desrosiers, Y. Dumas, and F. Soumis. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences*, 6:301–325, 1986.

O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194:229–237, 1999.

Y. Dumas, J. Desrosiers, and F. Soumis. The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54:7–22, 1991.

I. Dumitrescu. *Constrained Path and Cycle Problems.* PhD thesis, Department of Mathematics and Statistics, The University of Melbourne, May 2002.

D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

S. Gélinas, M. Desrochers, J. Desrosiers, and M.M. Solomon. A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research*, 61:91–109, 1995.

S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer, 2005.

S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. Technical Report G-2003-55, GERAD, Montreal, Canada, September 2003.

M. Jepsen, S. Spoorendonk, and B. Petersen. A branch-and-cut-and-price framework for VRP applied on CVRP and VRPTW. Master's thesis, DIKU Department of Computer Science, University of Copenhagen, August 2005.

N. Kohl, J. Desrosiers, O. B. G. Madsen, M.M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101 – 116, 1999.

H. Li and A. Lim. A metaheuristic for the pickup and delivery problem with time windows. In *The 13th IEEE International Conference on Tools with Artificial Intelligence, ICTAI-2001*, Dallas, USA, 2001.

Q. Lu and M. Dessouky. An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science*, 38:503–514, 2004.

J. Lysgaard. Reachability cuts for the vehicle routing problem with time windows. *European Journal of Operational Research*, 2005. To appear.

D. Naddef and G. Rinaldi. Branch-and-cut algorithms for the capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 3, pages 53–84. SIAM, Philadelphia, 2002.

G. Righini and M. Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. Technical report, Note del polo - Ricerca N.66, October 2004.

S. Ropke, J.-F. Cordeau, and G. Laporte. Models and a branch-and-cut algorithm for pickup and delivery problems with time windows. Technical report, CRT-2005-25, 2005.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows, 2004. Submitted to Transportation Science.

K.S. Ruland and E.Y Rodin. The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers and Mathematics with Applications*, 33(12):1–13, 1997.

M.W.P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29:17–29, 1995.

M.W.P. Savelsbergh and M. Sol. DRIVE: Dynamic routing of independant vehicles. *Operations Research*, 46:474–490, 1998.

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming)*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431, 1998.

M. Sigurd, D. Pisinger, and M. Sig. Scheduling transportation of live animals to avoid the spread of diseases. *Transportation Science*, 38:197–209, 2004.

M. Sol. *Column generation techniques for pickup and delivery problems*. PhD thesis, Technische Universitet Eindhoven, 1994.

P. Toth and D. Vigo. *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*. SIAM, Philadelphia, 2002.

L. A. Wolsey. *Integer Programming*. Wiley-Interscience series in discrete mathematics. Wiley-Interscience publication, 1998.

H. Xu, Z.-L. Chen, S. Rajagopal, and S. Arunapuram. Solving a practical pickup and delivery problem. *Transportation Science*, 37:347–364, 2003.

Part IV

Conclusion

Chapter 10

Conclusion

This thesis has touched many subjects within vehicle routing problems but the core problem, in the methods developed in the five papers, has always been a pickup and delivery problem with time windows (PDPTW). It is my hope that this thesis significantly has increased our knowledge about the PDPTW. The thesis has also moved the bounderies for which PDPTW/DARP instances that are solvable with exact methods and improved the quality of heuristics for the PDPTW. In the thesis

- We showed how the PDPTW can be used to model many other vehicle routing problems. This insight has allowed us to design a single heuristic that can solve many variants of vehicle routing problem, obtaining just as good solutions as specialized heuristics without the need for any retuning or customization of the heurisitic.
- We compared different models for the PDPTW in order to solve the problem to optimality.
- Several new valid inequalities and corresponding separation algorithms for the problem PDPTW and dial-a-ride problems (DARP) was proposed. We have not performed any polyhedral analysis, but computational experiments show that the valid inequalities are able to improve the lower bounds of the LP relaxations significantly.
- We saw that the models proposed in Chapter 8 combined with the new valid inequalities allowed us to solve some DARP instances up to 1000 times faster than in a recent study Cordeau [2006].
- We saw that the set partitioning formulations presented in Chapter 9 solved through column generation allowed us to solve larger PDPTW instances than was reported by Savelsbergh and Sol [1998].

The Adaptive Large Neighborhood search (ALNS) presented in Chapters 4 to 6 proved to be a worthwhile extension of the Large Neighborhood Search (LNS) heuristic proposed by Shaw [1998]. Using multiple, fast neighborhoods turned out to help the heuristic a great deal, and the adaptive mechanism was able to select a good weighting of the subheuristics. We believe that the heuristic can be applied to problems outside the vehicle routing domain as well. We expect it to be particularly well suited for tightly constrained problems where smaller neighborhoods can have trouble getting from one area of the solution space to another. We still need to substantiate this hypothesis by computational tests. Some implementation for the graph coloring problem is underway - even though this problem hardly can be considered as tightly constrained.

It would be interesting to combine the ALNS heuristic for the PDPTW with the exact methods to produce new heuristics. We could use the lower bound and exact methods for both the removal and insertion phase. To insert a set S of requests into a partial solution s using the exact methods one can follow this procedure

1. Create a graph G with nodes corresponding to the nodes of the PDPTW instance.
2. Add the edges occuring in the partial solution s.
3. Add all edges among nodes corresponding to requests in S
4. Add all edges to/from nodes in s from/to nodes corresponding to requests in S.
5. Perform preprocessing based on G, reducing time windows and removing infeasible arcs from G.
6. Solve the PDPTW on the graph G using one of the exact methods. The exact method can be truncated in different ways to speed up the solution time.

This would give optimal insertions if step 6 is solved to optimality, but it would also be quite time consuming, so it should be used rarely. It is similar to the approach presented by Franceschi et al. [2005] for the CVRP, but it can use any optimization method developed for the routing problem in step 6, and is not solving a general ILP, this could potentially mean quicker computations. The method could also be applied to insert edges instead of nodes.

The lower bounds developed could be used in a removal heuristic in the ALNS. In Chapter 4 we proposed the *worst removal*. This method defines a cost function $cost(i, s)$ for each request i in solution s and removes requests with high cost. The function $cost(i, s)$ could be redifined to the lower bound obtained when the edges adjacent to request i are fixed as they are in the solution s.

A third posibility is to use the exact methods as a postprocesing step after running the ALNS. While the ALNS is running a number of good, unique, solutions (e.g. 10) are stored. When the ALNS terminates a graph is created by taking the union of the edges used in the stored solution. The PDPTW is then solved to optimality on this graph and thus computes the optimal combination of the selected solutions. When the graph is sparse it should be reasonably fast to solve the problem through column generation. A similar strategy has been proposed and tested for the TSP by Cook and Seymour [2003] with worthwhile improvements in solution quality.

Some ideas for further improvements of the exact methods for the PDPTW based on set-partitioning formulations are given in the preface and in Chapter 9. Implementing (some of) these are my short term plans for the development of exact methods for the PDPTW, but the development possibilities certainly does not stop there. One project I personally would find intersting is to develop a pricing algorithm for the DARP that takes the ride time constraints into account. This would most like improve the branch-and-price results for the DARP instances if the pricing problem can be solved reasonably fast.

On a longer term horizon I hope to study exact methods for the VRPTW and/or CVRP. These are clearly the vehicle routing problems that are most “mature” when it comes to exact methods. I believe that there nevertheless still are room for improvements of the exact algorithms for both of the problems.

Chapter 11

Summary (in Danish)

Rutelægningsproblemer (eng. *vehicle routing problems)* er en vigtig klasse af optimeringsproblemer. I det grundlæggende problem er vi givet en række kunder med forskellige behov der skal forsynes med varer fra et depot. Til at forsyne kunderne har vi en flåde af køretøjer (typisk lastbiler) der hver har en begrænset laste kapacitet. Problemet består nu i at fremstille ruter der starter og slutter i depotet så hver kunde bliver besøgt præcist en gang og således af lastkapaciteten af lastbilen der betjener i ruten ikke overskrides - målet er at minimere omkostningen ved transporten. Omkostningen ved transporten udmåles ofte som den samlede distance kørt eller antallet af køretøjer der skal bruges til transporten.

En lang række varianter af problemet findes, f.eks. udvider nogle varianter problemet så hver kunde har et tidsvindue hvori betjeningen skal foregå og andre varianter definerer at varer skal samles op hos nogle kunder og returneres til depotet.

En vigtig variant af problemet er afhentnings og leveringsproblemet med tidsvinduer (eng. *pickup and delivery problem with time windows - PDPTW*). I dette problem er vi givet en række transportopgaver. Hver opgave består i at samle varer op på lokalitet A og bringe disse varer til lokalitet B. Der er knyttet et tidsvindue til både afhentningen og leveringen. Køretøjerne der benyttes til transporten starter og slutter deres rute på en *terminal* men de pålæsser og aflæsser ikke nødvendigvis varer her.

Rutelægningsproblemer er en vigtig klasse af problemer inden for både forskningen i operationsanalyse og i den virkelige verden. Inden for forskningen i operationsanalyse har rutelægningsproblemer udgjort en af de hyppigste problemtyper som nye varianter af metaheuristikker er blevet testet og sammenlignet på. Desuden, og måske mere vigtigt, er det at problemstillingen har været drivende for udviklingen af *branch-and-price* paradigmet.

Det er oplagt at problemet er anvendeligt i praksis. En lang række virksomheder bruger store summer på transport af varer vha. lastbiler. Hvis antallet af kørte kilometer kan bringes ned vil det give en besparelse på brandstofudgifterne og på udgifterne til vedligeholdelse af køretøjerne. Tilsvarende vil en reduktion i antallet af nødvendige køretøjer også give en besparelse. Toth and Vigo [2002b] skønner at brugen af computerbaserede løsningsmetoder til optimering af transportplanlægningsproblemer i industrien fører til besparelser på mellem 5% og 20% af transportomkostningerne, hvilket må forventes at være en betydelig sum i mange større virksomheder. En lang række virksomheder i Danmark bruger da også computerbaserede løsningsmetoder til den daglige planlægning af transportopgaver. Nogle eksempler er Arla, Statoil og Unicon (betonkørsel).

Hovedbidraget i afhandlingen er 5 artikler. Den første artikel beskriver en robust heuristik for en udvidet udgave af PDPTW. Heuristikken afprøves på en række standardproblemer fra litteraturen og denne test viser at heuristikken kan forbedre mange tidligere bedst kendte løsninger til standardproblemerne og heuristikken må betragtes som den bedste heuristik til PDPTW for tiden.

De næste to artikler foreslår en række forbedringer til heuristikken og viser hvordan en række standard ruteplanlægningsproblemer kan transformeres til et udvidet PDPTW og dermed løses ved hjælp af den allerede udviklede heuristik. Resultaterne er meget lovende idet heuristikken ofte løser disse problemer ligeså godt eller bedre end mere specialiserede heuristikker. Vi forbedrer igen en lang række løsninger til standard testproblemer for varianter af rutelægningsproblemet. På grund af disse positive erfaringer udleder vi essensen af heuristikken så den kan anvendes på optimeringsproblemer generelt.

De to sidste artikler omhandler også PDPTW, men denne gang anvendes eksakte optimeringsmetoder. Dvs. den løsning som returneres fra metoden er beviseligt den bedste der findes, givet de kriterier der optimeres efter. Da PDPTW er NP-hårdt er det beregningsmæssigt meget tungt at løse problemet eksakt, og for probleminstanser af selv moderat størrelse kan det være umuligt at finde den optimale løsning indenfor en overskuelig tidsperiode med en given løsningsmetode. De to artikler anvender to relaterede paradigmer til løsning af problemet (på engelsk *branch-and-cut* og *branch-and-cut-and-price*). De eksperimentelle resultater viser at løsningsmetoderne flytter grænserne for hvilke problemstørrelser der er mulige at løse til optimalitet. F.eks. ser vi i den ene artikel at løsningstiden for en probleminstans forbedres med mere end en faktor 1000 i forhold til en for nyligt offentliggjort løsningsmetode.

Udover de 5 artikler indeholder afhandling introducerende kapitler der beskriver vigtige rutelægningsproblemer, heuristikker og eksakte metoder.

Bibliography

R. Agarwal, R.K. Ahuja, G. Laporte, and Z.J. Shen. A composite very large-scale neighborhood search algorithm for the vehicle routing problem. In J. Y-T. Leung, editor, *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, pages 49–01 to 49–23. CRC Press, 2004.

R.K. Ahuja, Ö. Ergun, J.B. Ergun, and A.P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123:75–102, 2002.

E. Alba, editor. *Parallel Metaheuristics*. Wiley Series on Parallel and Distributed Computing. Wiley, 2005.

İ.K Altinel and T. Öncan. A new enhancement of the clarke and wright savings heuristic for the capacitated vehicle routing problem. *Journal of the Operational Research Society*, 56:954–961, 2005.

H. Ben Amor, J. Desrosiers, and A. Frangioni. Stabilization in column generation. Technical Report G-2004-62, GERAD, August 2004.

D. Applegate, R. Bixby, and V. Chvátal abd W. Cook. Implementing the dantzig-fulkerson-johnson algorithm for large traveling salesman problems. 97:91–153, 2003.

D. Applegate, W. Cook, S. Dash, and A. Rohe. Solution of a min-max vehicle routing problem. *INFORMS Journal on Computing*, 14(2):132–143, 2002.

T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34:209–219, 2006.

R. Bent and P. Van Hentenryck. A two-stage hybrid algorithm for pickup and delivery vehicle routing problem with time windows. *Computers & Operations Research*, 33(4):875–893, 2006.

U. Blasum and W. Hochstättler. Application of the branch and cut method to the vehicle routing problem. Technical Report zaik2000-386, Zentrum fur Angewandte Informatik Koln, 2000.

C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.

L.D. Bodin and T.R. Sexton. The multi-vehicle subscriber dial-a-ride problem. *TIMS Studies in Management Science*, 22:73–86, 1986.

N. Boland, J. Detridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, 2005. Forthcomming.

J. Bramel and D. Simchi-Levi. Set-covering-based algorithms for the capacitated vrp. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 4, pages 85–108. SIAM, Philadelphia, 2002.

O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, 39:104–118, 2005a.

O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, 39:119–139, 2005b.

A. Caprara, M. Fischetti, and P. Toth. Modeling and solving the train timetabling problem. *Operations Research*, 50(5):851–861, 2002.

A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 2005. Forthcoming.

N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, pages 315–338. Wiley, Chichester, UK, 1979.

G. Clarke and J.V. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

W. Cook and P. Seymour. Tour merging via branch-decomposition. *INFORMS Journal on Computing*, 15(3):233–248, 2003.

J.-F. Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54: 573–586, 2006.

J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M.M. Solomon, and F. Soumis. Vrp with time windows. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 7, pages 157–193. SIAM, Philadelphia, 2002.

J.-F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, and J.-S. Sormany. New heuristics for the vehicle routing problem. Technical Report G-2004-33, GERAD, Montreal, Canada, 2004.

J.-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30:105–119, 1997.

J.-F. Cordeau and G. Laporte. A tabu search algorithm for the site dependent vehicle routing problem with time windows. *INFOR*, 39:292–298, 2001.

J.-F. Cordeau and G. Laporte. The dial-a-ride problem (darp): Variants, modelling issues and algortihms. *4OR*, 1(2):89–101, 2003.

J.-F. Cordeau, G. Laporte, and A. Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52:928–936, 2000.

J.-F. Cordeau, G. Laporte, J.-Y. Potvin, and M.W.P. Savelsbergh. Transportation on demand. In C. Barnhart and G. Laporte, editors, *Transportation*, Handbooks in Operations Research and Management Science. Elsevier, Amsterdam, 2005a. Forthcoming.

J.-F. Cordeau, G. Laporte, M.W.P. Savelsbergh, and Daniele Vigo. Short-haul routing. In C. Barnhart and G. Laporte, editors, *Transportation*, Handbooks in Operations Research and Management Science. Elsevier, Amsterdam, 2005b. Forthcoming.

J.-C. Créput, A. Koukam, J. Kozlak, and J. Lukasik. An evolutionary approach to pickup and delivery problem with time windows. *Lecture Notes in Computer Science*, 3038:1102–1108, 2004.

G.B. Dantzig and J.H. Ramser. The truck dispatching problem. *Management Science*, 6:80–91, 1959.

G.B. Dantzig and P. Wolfe. Decompostion principle for linear programs. *Operations Research*, 8 (1):101–111, 1960.

G. Desaulniers, J. Desrosiers, A. Erdmann, M.M. Solomon, and F. Soumis. Vrp with pickup and delivery. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 9, pages 225–242. SIAM, Philadelphia, 2002.

G. Desaulniers, J. Desrosiers, I. Ioachim, M.M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T.G. Crainic and G. Laporte, editors, *Fleet management and logistics*, chapter 3, pages 57–93. Kluwer Academic Publishers, 1998.

G. Desaulniers, J. Desrosiers, and M.M. Solomon. Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 309–324. Kluwer, 2001.

J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis. Time constrained routing and scheduling. In C.L. Monma M.O. Ball, T.L. Magnanti and G.L. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, chapter 2, pages 35–139. North-Holland, Amsterdam, 1995.

J. Desrosiers, Y. Dumas, and F. Soumis. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American journal of mathematical and management science*, 6(3):301–315, 1986.

J. Desrosiers and M.E. Lübbecke. A primer in column generation. In G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors, *Column Generation*, volume 5 of *GERAD 25th Anniversary Series*, chapter 1, pages 1–32. Springer, 2005.

O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Applied Mathematics*, 194:229–237, 1999.

Y. Dumas, J. Desrosiers, and F. Soumis. The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54:7–22, 1991.

I. Dumitrescu. *Constrained Path and Cycle Problems*. PhD thesis, Department of Mathematics and Statistics, The University of Melbourne, May 2002.

I. Dumitrescu. Polyhedral results for the pickup and delivery travelling salesman problem. Technical Report CRT-2005-27, Centre for Research on Transportation, Canada, 2005.

A. H. G. Rinnooy Kan E. L. Lawler, J. K. Lenstra and D. B. Shmoys, editors. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley & Sons, 1985.

O. Ergun, J.B. Orlin, and A. Steele-Feldman. Creating very large scale neighborhoods out of smaller ones by compounding moves: A study on the vehicle routing problem. Technical Report MIT Sloan Working Paper No. 4393-02, MIT, October 2002.

D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44:216–229, 2004.

M.L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11:109–124, 1981.

P. M. França, M. Gendreau, G. Laporte, and F.M. Müller. The m-traveling salesman problem with minmax objective. *Transportation Science*, 29(3):267–275, 1995.

R. De Franceschi, M. Fischetti, and P. Toth. A new ilp-based refinement heuristic for vehicle routing problems. *Mathematical Programming, Ser. B*, 2005. Forthcomming.

R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, and R.F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming, Ser. A*, 2005. Forthcoming.

B. Funke, T. Grünert, and S. Irnich. Local search for vehicle routing and scheduling problems: Review and conceptual integration. *Journal of Heuristics*, 11:267–306, 2005.

H. Gehring and J. Homberger. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In *Proceedings of EUROGEN99.*, pages 57–64, 1999.

M. Gendreau, F. Guertin, J.-Y. Potvin, and R. Séguin. Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. Technical Report CRT-98-10, Centre for Research on Transportation, 1998.

M. Gendreau, M. Iori, G. Laporte, and S. Martello. A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. Technical report, CRT, September 2004.

M. Gendreau, G. Laporte, and J.-Y. Potvin. Metaheuristics for the capacitated vrp. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 6, pages 129–154. SIAM, Philadelphia, 2002.

M. Gendreau and J.-Y. Potvin. Metaheuristics in combinatorial optimization. *Annals of Operations Research*, 140:189–214, 2005.

A.M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study*, 2:82–114, 1974.

B.E. Gillet and L.R. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22:340–349, 1974.

B.L. Golden, E.A. Wasil, J.P. Kelly, and I-Ming Chao. The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets and computational results. In T.G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 33–56. Kluwer, Boston (MA), 1998.

M. Gronalt, R. F. Hartl, and M. Reimann. New savings based algorithms for time constrained pickup and delivery of full truckloads. *European Journal of Operational Research*, 151:520–535, 2004.

G. Gutin and A.P. Punnen, editors. *The Traveling Salesman Problem and Its Variations.* Kluwer Academic Publishers, 2002.

C.A. Hjorring. *The Vehicle Routing Problem and Local Search Metaheuristics.* PhD thesis, Department of Engineering Science, The University of Auckland, 1995.

D. Huisman, R. Jans, M. Peters, and A.P.M. Wagelmans. Combining column generation and lagrangian relaxation. In G. Desaulniers, Jacques Desrosiers, and M.M. Solomon, editors, *Column Generation*, GERAD 25th Anniversary Series, chapter 9, pages 247–270. Springer, 2005.

A. Imai, E. Nishimura, and S. Papadimitriou. Berth allocation with service priority. *Transportation Research Part B*, 37:437–457, 2003.

G. Ioannou, M. Kritikos, and G. Prastacos. A greedy look-ahead heuristic for the vehicle routing problem with time windows. *Journal of the Operational Research Society*, 52:523–537, 2001.

M. Iori, J.J. Salazar González, and D. Vigo. An exact approach for the vehicle routing problem with two-dimensional loading constraints. Technical Report OR-04-6, Universita di Bologna, 2004.

S. Irnich, B. Funke, and T. Grünert. Sequential search and its application to vehicle-routing problems. *Computers & Operations Research*, 2005. Forthcoming.

S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. Technical Report G-2003-55, GERAD, Montreal, Canada, September 2003.

J.-J. Jaw, A.R. Odoni, H.N. Psaraftis, and N.H.M. Wilson. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B*, 20(3):243–257, 1986.

M. Jepsen, S. Spoorendonk, and B. Petersen. A branch-and-cut-and-price framework for vrp applied on cvrp and vrptw. Master's thesis, DIKU Department of Computer Science, University of Copenhagen, August 2005.

B. Kalantari, A.V. Hill, and S.R. Arora. An algorithm for the traveling salesman problem with pickup and delivery problems. *European Journal of Operational Research*, 22:377–386, 1985.

B. Kallehauge and N. Boland. Path inequalities for the vehicle routing problem with time windows. Technical report, Centre for Traffic and Transport, Technical University of Copenhagen, May 2005.

B. Kallehauge, J. Larsen, and O.B.G. Madsen. Lagrangian duality applied to the vehicle routing problem with time windows. *Computers & Operations Research*, 33(5):1464–1487, 2006.

H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Number ISBN: 3-540-40286-1. Springer, 2004.

N. Kohl. *Exact methods for time constrained routing and related scheduling problems*. PhD thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 1995.

N. Kohl, J. Desrosiers, O.B.G. Madsen, M.M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116, 1999.

J. Kytöjoki and O. Bräysy. Huge scale vehicle routing problem solving with efficient metaheuristics. Presentation, ROUTE2005, Bertinoro, Italy, June 2005.

G. Laporte and F. Semet. Classical heuristics for the capacitated vrp. In Paulo Toth and Daniele Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 5, pages 109–128. SIAM, Philadelphia, 2002.

H.C. Lau and Z. Liang. Pickup and delivery with time windows : Algorithms and test case generation. In *The 13th IEEE International Conference on Tools with Artificial Intelligence, ICTAI-2001*, pages 333–340, Dallas, USA, 2001.

F. Li, B. Golden, and E.Wasil. Very large-scale vehicle routing: new test problems, algorithms, and results. *Computers & Operations Research*, 32:1165–1179, 2005.

H. Li and A. Lim. A metaheuristic for the pickup and delivery problem with time windows. In *The 13th IEEE International Conference on Tools with Artificial Intelligence, ICTAI-2001*, pages 160–170, Dallas, USA, 2001.

S. Lin and B.W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

F.-H. Liu and S.-Y. Shen. A method for vehicle routing problems with multiple vehicle types and time windows. *Proceedings of Natural Science Council*, 23(4):526–536, 1999.

Q. Lu and M. Dessouky. An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science*, 38(4):503–514, 2004.

Q. Lu and M.M. Dessouky. A new insertion-based construction heuristic for solving the pickup and delivery problem with hard time windows. *European Journal of Operational Research*, 2005. Forthcoming.

M. Lübbecke. *Engine scheduling by column generation*. PhD thesis, Techniscen Universtät Braunschweig, 2001.

M.E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 2005. Forthcomming.

J. Lysgaard. Reachability cuts for the vehicle routing problem with time windows. *European Journal of Operational Research*, 2005. Forthcoming.

J. Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming, Ser. A*, 100:423–445, 2004.

D. Mester and O. Bräysy. Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers & Operations Research*, 32:1593–1614, 2005.

S. Mitrović-Minić. Pickup and delivery problem with time windows: A survey. Technical Report SFU CMPT TR 1998-12, Simon Fraser University, 1998.

D. Naddef and G. Rinaldi. Branch-and-cut algorithms for the capacitated vrp. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 3, pages 53–84. SIAM, Philadelphia, 2002.

W.P. Nanry and J.W. Barnes. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B*, 34:107–121, 2000.

G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, New York, 1988.

I. H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41:421–251, 1993.

A. Oukil, H. El Gueddari, J. Desrosiers, and H. Ben Amor. Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems. Technical Report G-2004-75, GERAD, October 2004.

G. Pankratz. A grouping genetic algorithm for the pickup and delivery problem with time windows. *OR Spectrum*, 27:21–41, 2005.

M.B. Pedersen. *Optimization models and solution methods for intermodal transportation*. PhD thesis, Centre for Traffic and Transport, Technical University of Denmark, 2005.

M. Poggi de Aragão and E. Uchoa. Integer program reformulation for robust branch-and-cut-and-price algorithms. November 2003.

J.-Y. Potving and J.-M. Rousseau. An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, 46:1433–1446, 1995.

C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31:1985–2002, 2004.

H.N. Psaraftis. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14(2):130–154, 1980.

T.K. Ralphs and M.V. Galati. Decomposition in integer linear programming. In John K Karlof, editor, *Integer Programming: Theory and Practice*, volume 3 of *Operations Research Series*. CRC Press, 2005.

G. Righini and M. Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. Technical report, Note del polo - Ricerca N.66, October 2004.

G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. Technical Report Note del polo - Ricerca N.69, Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano, January 2005.

S. Ropke and J.-F. Cordeau. Branch-and-cut-and-price for the pickup and delivery problem with time windows. Working Paper, 2006.

L.-M. Rousseau, M. Gendreau, and D. Feillet. Interior point stabilization for column generation. Technical Report C7PQMR PO2003-39-X, Centre for Research on Transportation, 2003.

K.S. Ruland and E.Y Rodin. The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers and Mathematics with Applications*, 33(12):1–13, 1997.

R. A. Russell. Hybrid heuristics for the vehicle routing problem with time windows. *Transportation Science*, 29(2):156–166, 1995.

D.M. Ryan, C. Hjorring, and F. Glover. Extensions of the petal method for vehicle routeing. *Journal of the Operational Research Society*, 44(3):289–296, 1993.

M.W.P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29:17–29, 1995.

M.W.P. Savelsbergh and M. Sol. DRIVE: Dynamic routing of independent vehicles. *Operations Research*, 46:474–490, 1998.

A. Schrijver. On the history of combinatorial optimization (till 1960). In K. Aardal, G.L. Nemhauser, and R. Weismantel, editors, *Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, chapter 1. Elsevier, 2005.

G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171, 2000.

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming)*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431, 1998.

M.M. Sigurd. *Column Generation Methods and Applications*. PhD thesis, Dept. of Computer Science, University of Copenhagen, Denmark, June 2004.

M.M. Sigurd, D. Pisinger, and M. Sig. Scheduling transportation of live animals to avoid the spread of diseases. *Transportation Science*, 38:197–209, 2004.

M.M. Sigurd and D.M. Ryan. Stabilized column generation for set partitioning optimisation. Technical report, Dept. of Engineering Science, University of Auckland, 2003.

M.M. Sigurd, N.L. Ulstein, B. Nygreen, and D.M. Ryan. Ship scheduling with recurring visits and visit separation requirements. In G. Desaulniers, Jacques Desrosiers, and M.M. Solomon, editors, *Column Generation*, GERAD 25th Anniversary Series, chapter 8, pages 225–245. Springer, 2005.

M. Sol. *Column generation techniques for pickup and delivery problems.* PhD thesis, Technische Universitet Eindhoven, 1994.

M.M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

P.M. Thompson and J.B. Orlin. The theory of cyclic transfers. Technical Report MIT Operations Research Center Report 200-89, MIT, 1989.

P. Toth and D. Vigo. Heuristic algorithms for the handicapped person transportation problem. *Transportation Science*, 31(1):60–71, 1997.

P. Toth and D. Vigo. A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls. *European Journal of Operational Research*, 113:528–543, 1999.

P. Toth and D. Vigo. Branch-and-bound algorithms for the capacitated vrp. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 2, pages 29–51. SIAM, Philadelphia, 2002a.

P. Toth and D. Vigo. An overview of vehicle routing problems. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 1, pages 1–26. SIAM, Philadelphia, 2002b.

P. Toth and D. Vigo. The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing*, 15(4):333–346, 2003.

F. Vanderbeck. On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48(1):111–128, 2000.

V.V. Vazarani. *Approximation Algorithms.* Springer-Verlag, Berlin, 2001.

Stefan Voß. Meta-heuristics: The state of the art. *Lecture Notes in Computer Science*, 2148, 2001.

L.A. Wolsey. *Integer Programming.* Wiley-Interscience series in discrete mathematics. Wiley-Interscience publication, 1998.

H. Xu, Z.-L. Chen, S. Rajagopal, and S. Arunapuram. Solving a practical pickup and delivery problem. *Transportation Science*, 37:347–364, 2003.