Integrated Optimization of Vehicle and Crew Scheduling in Public Transport

Govinda Raja Perumal, Shyam Sundar

Publication date:
2019

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Integrated Optimization of Vehicle and Crew Scheduling in Public Transport

Shyam Sundar Govinda Raja Perumal

Kongens Lyngby 2019
Public transportation is recognized as a crucial backbone for sustainable urban development since it enhances mobility by providing infrastructure and services for the efficient movement of people. The primary focus of this thesis is on improving the efficiency of bus services from the perspective of transport companies. Maintenance and fuel consumption of buses and the wages paid to bus drivers are the main factors that contribute to the total operational cost. In particular, the cost of the crew is approximately 60% of the total operational cost. Transport companies are challenged to create cost-effective vehicle and crew schedules for cities with large-scale transport systems. Several practical conditions such as the infrastructure properties and labor regulations that govern the working conditions of bus drivers have to be considered during the operational planning process. The transport companies are also affected by the initiatives taken by the European Union to reduce greenhouse gas emissions.

Providing a bus service involves solving several planning problems such as line planning, timetabling, vehicle scheduling and crew scheduling. These problems are traditionally solved sequentially. However, in recent years, there has been growing research on integrating two or more problems. This thesis studies the impact of integrating the vehicle and crew scheduling problems. Simultaneously handling the vehicle and crew scheduling aspects could potentially improve the operational efficiency of transport systems.

This thesis aims to develop optimization algorithms based on operations research (OR) methods that minimize the total operational cost while handling all practical complexities of a large-scale transport system. One practical complexity that is incorporated in this thesis is the use of staff car by bus drivers. A staff
car is a company-owned car that is used to increase flexibility when scheduling bus drivers. In most cases, staff cars are necessary for transport companies to find a feasible crew schedule. This thesis also incorporates the limited driving range and recharging requirements of electric buses. An integrated approach is proposed that solves the electric vehicle and crew scheduling problems simultaneously. Real-world instances from several Northern European transport companies were obtained to test the developed optimization algorithms. The integrated approach decreases the total operational cost by 1.17-4.37% on average when compared to the traditional sequential approach.

This thesis is carried out in collaboration with industrial partner QAMPO ApS to incorporate the developed algorithms in a decision support system that aids transport companies in implementing cost-effective vehicle and crew schedules. Therefore, this study aims to create a significant positive impact on the transport industry and also adds value to the current state-of-the-art knowledge on utilizing OR-based methods for integrated transport planning problems.
Summary (Danish)


Dette ph.d.-projekt sigter mod at udvikle optimeringsalgoritmer baseret på ope-

Denne afhandling er gennemført i samarbejde med industripartneren QAMPO ApS for derigennem at integrere de udviklede algoritmer i et beslutningsstøttesystem, der hjælper transportvirksomheder i implementering af systemerne. Derfor sigter denne afhandling mod at gøre det muligt for transportvirksomheder at blive endnu mere effektive og bedre til at planlægge deres ressourcer. Samtidig tilføjer afhandlingen også verdi til den state-of-art om anvendelse af OR-baserede metoder til integrerede transportplanlægningsproblemer.
This thesis has been submitted to the Department of Technology, Management and Economics, Technical University of Denmark (DTU) in partial fulfillment of the requirements for acquiring the Doctor of Philosophy (PhD) degree. The PhD project has been carried out during the period between January 2017 and December 2019.

The thesis deals with improving the efficiency of bus services using operations research methods. The study has been conducted in collaboration with QAMPO ApS, Denmark under the Industrial PhD programme. QAMPO ApS is a decision science company that focus on developing and deploying solution methods to public bus transport companies. The PhD project has been supervised by Associate Professor Richard Martin Lusby, DTU and co-supervised by Professor Jesper Larsen, DTU and Morten Riis, QAMPO ApS.

The thesis consists of two parts. The first part gives an introduction and overview of the research area. The second part consists of three scientific papers that have been written during the PhD project.

Lyngby, 31 December 2019

Shyam Sundar Govinda Raja Perumal
Acknowledgements

During the past three years, I had the greatest opportunity to learn from the most talented people in academia and the industry which has been the most rewarding experience of the PhD project. It would not have been possible to complete it without their invaluable support and contribution.

I would like to thank my supervisors at DTU, Associate professor Richard Lusby and Professor Jesper Larsen, for their continuous guidance and patience. Their dedicated assistance with the various aspects of the PhD process including proof-reading and providing constructive feedback proved to be very helpful. Most importantly, they suggested the right directions of research when new and challenging problems emerged from the industry during the project. It was a very engaging learning and working experience with them.

I had the opportunity to work with Professor Dennis Huisman and Associate professor Twan Dollevoet during my four-month research stay at the Econometric institute at Erasmus University, Rotterdam. I would like to thank them for hosting me and providing an excellent work environment. Their valuable suggestions and feedback are very appreciated. Dennis’s extensive knowledge on public transport was a huge source of inspiration and I would like to express my gratitude for his exceptional hospitality.

I would also like to recognise my fellow PhD students at DTU and Erasmus University as I enjoyed the discussions about their respective research areas.

I must thank Morten Riis from QAMPO ApS primarily for providing practical guidance to help broaden my understanding of the public transport industry. He
greatly helped in steering the research project in directions that are of significant relevance to the industry. I would like to thank the development teams of QAMPO and Trapeze Group Europe for their support. Particularly, I would like to thank Tue Christensen from QAMPO and Kasper Sørensen from Trapeze for providing sufficient data for research and valuable insights into the planning process in the public transport industry.

I would like to acknowledge Ali Khatam from QAMPO for initiating the collaboration between DTU and QAMPO for this industrial PhD project. I am very grateful to him for motivating me to tackle new challenges. I also thank Innovation Fund Denmark for financially supporting this project.

Furthermore, I would like to thank my friends who have always supported me. In particular, I would like to thank my roommates during the initial stages of this PhD, Joseph and Sowjanya, for their support. My parents had shown great interest in the project and, of course, cared for my well-being. I thank them for their constant encouragement and emotional support during the PhD study. Finally, I would like to thank my wife, Divya. She had not only been patient but was a pillar of support and a source of positive energy during the final stages of the study.
Contents

Summary (English) i
Summary (Danish) iii
Preface v
Acknowledgements vii

I Background and Problem Description 1

1 Introduction 3
  1.1 Public Transportation Planning Process ................. 4
  1.2 Motivation and Purpose of Thesis .................. 6
  1.3 Contribution of Thesis .......................... 8
  1.4 Thesis Outline ................................ 9

2 Combinatorial Optimization Problems 11
  2.1 Set Partitioning/Covering Problem ......................... 11
  2.2 Shortest Path Problem with Resource Constraints ........ 12

3 Solution Methods 15
  3.1 Column Generation ................................ 15
    3.1.1 Integer Solutions ................................ 17
  3.2 Dynamic Programming ................................ 19
  3.3 Adaptive Large Neighborhood Search .................. 21
## 4 Vehicle and Crew Scheduling: Models, Methods and Applications 25

4.1 Vehicle Scheduling Problem ........................................ 26
   4.1.1 Single Depot Vehicle Scheduling Problem ................. 26
   4.1.2 Multiple Depot Vehicle Scheduling Problem ............ 27
   4.1.3 Electric Vehicle Scheduling Problem .................... 31
4.2 Crew Scheduling Problem ........................................ 36
   4.2.1 Labor Regulations ........................................... 36
   4.2.2 Mathematical Model ......................................... 38
   4.2.3 Solution Methods ............................................ 39
4.3 Integrated Vehicle and Crew Scheduling Problem .............. 40
   4.3.1 Mathematical Model ......................................... 42
   4.3.2 Solution Methods ............................................ 43
4.4 Rescheduling and Robustness ..................................... 46
4.5 Other Future Research Areas ..................................... 48

## 5 Discussion and Conclusion 49

5.1 Overview of Papers .............................................. 49
5.2 Impact in Practice ............................................... 50
5.3 Future Work ..................................................... 52

## II Scientific Papers 67

A A Matheuristic for the Driver Scheduling Problem with Staff Cars 69
   A.1 Introduction .................................................. 70
   A.2 Related Literature ............................................ 74
   A.3 Problem Description and Mathematical Modelling ........... 77
   A.4 Solution Method ............................................... 80
      A.4.1 Greedy heuristic ....................................... 80
      A.4.2 Matheuristic ............................................. 82
   A.5 Computational Study .......................................... 89
      A.5.1 Instances .................................................. 89
      A.5.2 Parameter setup ......................................... 91
      A.5.3 Performance of destroy and repair methods ............ 92
   A.6 Conclusion .................................................... 98

B A Column Generation Approach for the Driver Scheduling Problem with Staff Cars 105
   B.1 Introduction .................................................. 106
   B.2 Related Literature ............................................ 107
<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.3 Problem Description</td>
<td>111</td>
</tr>
<tr>
<td>B.3.1 Labor regulations</td>
<td>111</td>
</tr>
<tr>
<td>B.3.2 Staff cars</td>
<td>113</td>
</tr>
<tr>
<td>B.3.3 Mathematical model</td>
<td>114</td>
</tr>
<tr>
<td>B.4 Methodology</td>
<td>116</td>
</tr>
<tr>
<td>B.4.1 Piece of work generation</td>
<td>116</td>
</tr>
<tr>
<td>B.4.2 Shortest path problem with resource constraints</td>
<td>118</td>
</tr>
<tr>
<td>B.4.3 Staff car subproblem</td>
<td>124</td>
</tr>
<tr>
<td>B.4.4 Integer solutions</td>
<td>125</td>
</tr>
<tr>
<td>B.5 Computational Study</td>
<td>128</td>
</tr>
<tr>
<td>B.5.1 Instances</td>
<td>128</td>
</tr>
<tr>
<td>B.5.2 Results</td>
<td>129</td>
</tr>
<tr>
<td>B.5.3 Analysis of domination rules in the SPPRC</td>
<td>132</td>
</tr>
<tr>
<td>B.5.4 Sensitivity analysis</td>
<td>135</td>
</tr>
<tr>
<td>B.6 Discussion and Conclusion</td>
<td>136</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solution Approaches for Vehicle and Crew Scheduling with Electric Buses</th>
<th>143</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.1 Introduction</td>
<td>144</td>
</tr>
<tr>
<td>C.2 Related Literature</td>
<td>147</td>
</tr>
<tr>
<td>C.3 Problem Description</td>
<td>150</td>
</tr>
<tr>
<td>C.4 Mathematical Formulation</td>
<td>152</td>
</tr>
<tr>
<td>C.5 Lower Bounds and Fast Upper Bounds</td>
<td>155</td>
</tr>
<tr>
<td>C.5.1 Integrated approach</td>
<td>156</td>
</tr>
<tr>
<td>C.5.2 Independent approach</td>
<td>157</td>
</tr>
<tr>
<td>C.5.3 Sequential approach</td>
<td>157</td>
</tr>
<tr>
<td>C.6 Adaptive Large Neighborhood Search</td>
<td>158</td>
</tr>
<tr>
<td>C.6.1 Neighborhoods</td>
<td>160</td>
</tr>
<tr>
<td>C.7 Computational Study</td>
<td>163</td>
</tr>
<tr>
<td>C.7.1 Instances</td>
<td>163</td>
</tr>
<tr>
<td>C.7.2 Results of independent and integrated approaches</td>
<td>164</td>
</tr>
<tr>
<td>C.7.3 Results of sequential approach</td>
<td>166</td>
</tr>
<tr>
<td>C.7.4 Experimental setup of ALNS</td>
<td>167</td>
</tr>
<tr>
<td>C.7.5 Results of ALNS</td>
<td>168</td>
</tr>
<tr>
<td>C.7.6 Analysis of neighborhoods</td>
<td>172</td>
</tr>
<tr>
<td>C.7.7 Sensitivity analysis of electric bus technology</td>
<td>174</td>
</tr>
<tr>
<td>C.8 Conclusion</td>
<td>176</td>
</tr>
</tbody>
</table>
Part I

Background and Problem Description
In 2018, the United Nations (UN) reported that 55% of the world population reside in urban areas, which is estimated to be 4.2 billion people (United Nations, 2018). By 2050, 68% of the world population is projected to be urban. The growth of world population and urbanization demand building sustainable cities that provide opportunities for social and economic development while reducing adverse impacts on the environment. Public transportation is recognized as a crucial backbone for sustainable urban development since it enhances mobility by providing infrastructure and services for safe and efficient movement of people. A sustainable transport system prevents severe traffic congestion, road accidents, air and noise pollution. However, planning, operating and controlling a public transport system of a city is known to be challenging due to its sheer size and complexity. Several stakeholders namely public authorities, public transport companies and users or passengers, with different goals are involved in the transport planning process. The passengers usually have varying socio-economic characteristics and expect a high level of service; i.e. the transport system should be safe, accessible, comfortable, affordable and should provide the possibility of reaching destinations quickly. The objective of transport companies is to provide high quality service to the passengers while minimizing their overall operational cost (Desaulniers and Hickman, 2007 and Ibarra-Rojas et al., 2015). A public transport system is typically designed with multiple modes of transport such as tram, metro, train and bus, and the aim of the system is to seamlessly integrate the different services for a better passenger experience.
In this thesis, the focus is on improving the efficiency of bus services from the transport companies’ perspective. For a transport company, maintenance and fuel consumption of buses and the wages paid to bus drivers are the main factors that contribute to the operational cost. In particular, the cost of the crew is approximately 60% of the total operational cost. Reducing the operational cost directly influences the ridership cost and increases attractiveness of bus transportation for the passengers. However, transport companies are challenged to create cost-effective bus and driver schedules for cities with large-scale transport systems. Several practical conditions such as the labor regulations that govern the working conditions of bus drivers and infrastructure properties have to be considered during the operational planning process. The transport companies and the industry in general are also affected by the climate agenda initiated by government and intergovernmental organizations. In accordance with the UN Paris Agreement (United Nations Climate Change, 2015), the European Union (EU) aims to create a climate-neutral economy by 2050 (European Union, 2018). Therefore, the EU has initiated strategies to reduce greenhouse emission, which also includes the modernization of the transport infrastructure.

1.1 Public Transportation Planning Process

Providing a bus service involves several stages of planning. They range from making long-term decisions such as investment in infrastructure to short-term decisions of executing day-to-day operations. The entire planning process of public transportation is computationally intractable and cannot be solved in one integrated step. Hence, it is divided into several problems which are solved in a sequential manner as shown in Figure 1.1. The different planning problems are discussed in Desaulniers and Hickman (2007), Schöbel (2012) and Ibarra-Rojas et al. (2015). One would also find the planning process to be similar to other transport industries such as railways (see e.g. Lusby et al., 2018). The *infrastructure* is represented as a public transportation network that describes the streets and bus stops of a city. In a tram or a railway system, the network represents the track system. A *line* is defined as a path or a route in the city along which a bus service is offered and the *frequency* of a line says how often the service is offered along the line within a given time period (e.g. one hour). The *line planning* and *frequency setting problem* determine the lines and the frequencies of the lines based on forecast passenger demand. The frequency setting problem also takes the demand patterns during different periods (morning, afternoon, evening) of operation into account. *Timetabling* is the process of defining arrival and departure times at all bus stops in the city network in order to meet the given frequency and level of service of each line. A timetable corresponds to a set of *trips* with arrival and departure bus stops and times.
The vehicle scheduling problem assigns buses to the timetabled trips such that every trip is covered by a bus and the objective is to minimize the operational cost based on bus usage. In a bus transportation setting, only one type of crew, i.e. the bus drivers, is required for performing the services, whereas drivers, conductors and catering staff are required in a train or airline setting. A duty is defined as the work of a bus driver for a day and the crew scheduling problem is concerned with determining sets of duties to cover all scheduled trips of vehicles. The objective of the crew scheduling problem is to minimize total wages paid to the drivers and the duties are subject to a wide range of labor union rules and regulations. The crew rostering problem consists of constructing and assigning weekly or monthly work schedules (called rosters) from the generic daily duties to the available drivers. The validity of the rosters is also restricted by labor union rules and regulations. During operation of transport systems, uncertain elements such as vehicle breakdowns or extreme weather conditions can severely disrupt the planned activities of vehicles and crew. Recovery plans and real-time control strategies are often implemented to reduce the impact of disruptions.

Figure 1.1: Public Transportation Planning Process.

Figure 1.1 gives an overview of the different problems in public transportation at the strategic, tactical and operational stages. The figure also indicates an estimate of the time each planning stage is considered before the day-of-operation. The infrastructure is rarely changed and potentially remains the same for many
years. The timetabling process is typically carried out a year in advance and the timetables are known to be different on weekdays, weekends and public holidays. Public authorities often are responsible for the timetabling process where the emphasis is on passenger service and the objective, most commonly, is to minimize travel or transfer times for passengers. The transport companies construct vehicle and crew schedules for the different timetables. Desrochers and Soumis (1989) state that a transport company in Montreal, Canada usually used a crew schedule for about half a year. At NS, the largest passenger railway operator in the Netherlands, the crew schedules for the annual plan are initially constructed and they are modified six times a year if there are specific changes in the timetable and vehicle schedule for a particular day (Huisman, 2007). Figure 1.1 is similar to the figure presented in Lusby et al. (2018). However, some authors (e.g. Ibarra-Rojas et al., 2015) have placed the vehicle and crew scheduling problems at the operational planning stage. Furthermore, an additional planning stage called control is included to describe the real-time control strategies and disruption management of transport systems.

1.2 Motivation and Purpose of Thesis

In most countries, crew is an expensive resource and optimizing the utilization of crew is of vital importance to all transport companies. Various complicating factors such as the labor regulations make the process of finding a cost-effective crew schedule difficult. Operations research (OR) has successfully been applied for solving a wide range of optimization problems in the public transport industry. The crew scheduling problem has been studied since the 1980s due to its particularly hard nature. Several commercial automated planning systems based on OR methods have been developed over the years to aid transport companies in finding cost-effective crew schedules while respecting the different labor regulations.

This PhD project is carried out in collaboration with QAMPO ApS, which is a decision science company that focuses on developing optimization algorithms for solving several planning and scheduling problems in various industries such as transportation and healthcare. One of QAMPO’s strategic partners is Trapeze Group Europe A/S, an international provider of decision support systems to transport companies. The partnership between the two firms aims to fulfill the business needs of transport companies. However, due to the growing size of transport systems and rapidly changing policies, transport companies are constantly seeking for techniques that can improve efficiency of their operations while managing the various complexities. The cost of crew contributes to approximately 60% of a medium-sized transport company’s operational expense in
1.2 Motivation and Purpose of Thesis

Northern Europe. For a transport company in Denmark with over 600 drivers, it was shown that a cost reduction of 1.2% represents 2-2.5 million DKK (301,117 €) in annual savings. Therefore, a small improvement in the cost of a crew schedule can lead to large savings.

The primary motivation of this PhD thesis is to develop and implement state-of-the-art optimization algorithms based on OR methods, which can capture all the complexities that arise in practice and improve efficiency of transport systems. Consequently, the developed methods are to be built as part of a decision support system so that transport companies can utilize the solutions provided by the methods.

In the public transport industry, as illustrated in Figure 1.1, the transportation problems are traditionally solved in a sequential manner. However, there has been an increasing focus on integrating two or more planning problems in recent years. At the tactical planning stage, vehicle and crew scheduling are the primary drivers of cost, and the traditional sequential approach finds the vehicle schedule first before finding the crew schedule. Simultaneously handling the vehicle and crew scheduling aspects could potentially lead to further cost reductions and efficiency gains for transport systems. However, the integrated vehicle and crew scheduling problem is more complex to formulate and requires tremendous computational effort to solve. The increased computational complexity of the integrated approach is one of the main reasons for practitioners to exploit a sequential approach. The transport companies also find computation time as an important criterion during the planning process. Short computation times to compute vehicle and crew schedules enables quick implementation of the schedules in practice and offers additional flexibility to analyze different scenarios. Therefore, the aim of this thesis, as illustrated in Figure 1.2, is also to study and integrate the vehicle and crew scheduling problem for large-scale transport systems.

Figure 1.2: The PhD project aims to integrate the vehicle and crew scheduling problems which are solved sequentially in the public bus transport industry. By integrating the two scheduling problems, the efficiency of the transport systems in terms of operational cost is expected to be improved.
1.3 Contribution of Thesis

The PhD project has primarily focused on the industrial application of the developed optimization algorithms. Therefore, the scalability of the algorithms in an industrial setting has been one of the most crucial tasks that had to be considered in this project. It is known from the OR literature that solving the vehicle and crew scheduling problems in an integrated manner would improve efficiency of transport systems. However, the challenge of this project was to devise an optimization algorithm based on OR methods that can handle all the practical complexities of a large-scale transport system and computes high quality solutions in reasonable computation times. Therefore, the following two research questions were initially formulated:

- **How could large-scale vehicle and crew scheduling problems be integrated using optimization algorithms?**

- **What is the overall benefit, in terms of operational cost, of integrating vehicle and crew scheduling problems?**

In this thesis, large-scale real-life instances from several transport companies in Northern Europe are considered for integration of the two scheduling problems. The largest instance considered for the integrated vehicle and crew scheduling problem contains 1,109 timetabled trips. Furthermore, the study incorporates numerous labor regulations into the crew scheduling process. In relation to the second research question, the thesis shows that integrated approaches can lead to significant improvements over a traditional sequential approach.

During the project, other practical conditions that transport companies consider in the scheduling process were encountered. One of the special constraints is the use of staff cars by the bus drivers. A staff car is a company-owned car that is utilized to increase flexibility when scheduling the bus drivers. In the crew scheduling process, the infrastructure properties of the city network such as the locations of the driver depots and canteen facilities for the drivers have to be considered. Staff cars are often necessary for most transport companies to respect the crew rules and the infrastructure properties.

The other special constraint is related to the electrification of bus fleets, which must be considered by 2025. Electric buses provide significant environmental benefits, but they are less flexible than the conventional diesel buses due to their limited driving range and longer recharging times. Electric buses provide additional operational challenges and consequently, the locations of recharging stations in the city network have to be considered for an efficient operation
of a zero-emission transportation system. Therefore, given the present and imminent challenges in the public bus industry, an additional research question is formulated in this study:

- **What is the impact of incorporating real-life complexities in optimization algorithms?**

The overall contributions of this thesis can be summed up in two points. Firstly, by solving the vehicle and crew scheduling problem using an integrated approach, this thesis adds value to the current state-of-the-art knowledge on utilizing OR based methods for integrated planning. The second point is regarding the inclusion of real-life complexities such as the use of staff cars, electric vehicles and the infrastructure properties of a city network. Since the study aims to make an impact on the public transport industry by providing managerial insights into the different planning problems, the inclusion of the practical conditions could be seen as a major contribution of this thesis. Furthermore, essential parts of the developed algorithms have been integrated into existing planning tool that is utilized by several transport companies. Therefore, the applicability of the developed algorithms can be stated as the practical contribution of this thesis.

### 1.4 Thesis Outline

This thesis is divided into two parts. Part I describes the theoretical background and literature review of the vehicle and crew scheduling problems in the public transport industry. Part II presents the three scientific papers that are the main outcomes of the work done during this PhD project.

Part I is made up of five chapters that includes this introductory chapter. Chapters 2 and 3 introduce and discuss the combinatorial optimization problems and the solutions methods that have been extensively used in this thesis, respectively. Chapter 4 gives a detailed literature review of the models and solution methods to solve the vehicle and crew scheduling problems. Finally, Chapter 5 gives an overview of the three scientific papers and summarizes the contributions of this thesis.

Part II is the collection of the three scientific papers. Papers A and B are related to the usage of staff cars in the crew scheduling problem. Paper C is related to the integration of electric vehicle and crew scheduling problems.
A combinatorial optimization problem is concerned with finding an optimal solution in a finite set of feasible solutions. An objective function to minimize (or maximize) is used for evaluating the quality of a solution. In this chapter, two well-known combinatorial optimization problems that are used in this thesis are introduced.

2.1 Set Partitioning/Covering Problem

Given a finite set of elements $M$, a finite set $N$ of subsets of $M$ and a cost associated with each member in $N$, the set partitioning problem (SPP) is defined as the problem of finding a minimum cost subset of $N$ that is a partition of $M$ (Balas and Padberg, 1976).

A binary matrix $A$ is defined, where $a_{ij}$ is equal to 1 if a subset $j \in N$ contains element $i \in M$ and 0 otherwise. Furthermore, each subset $j \in N$ has a cost $c_j$. Let $x_j$ be a binary decision variable that equals 1 if subset $j \in N$ is part of the
solution and 0 otherwise. The SPP is formulated as follows:

\[
\text{Minimize } \sum_{j \in N} c_j \cdot x_j
\]  

subject to,

\[
\sum_{j \in N} a_{ij} \cdot x_j = 1 \quad \forall i \in M
\]  
\[
x_j \in \{0, 1\} \quad \forall j \in N
\]

The set covering problem (SCP) is also related to finding a subset of \( N \) at a minimum cost. However, the SCP is a relaxation of the SPP, where each element \( i \in M \) can be in more than one subset \( j \in N \) in the solution instead of in exactly one. The SCP formulation can be obtained by replacing the equality signs in constraints (2.2) with “\( \geq \)” signs. Both the SPP and the SCP are known to be \( \mathcal{NP} \)-hard (Garey and Johnson, 1979).

The SPP/SCP models have been extensively used to solve vehicle and crew scheduling problems. Some examples include Desrochers and Soumis (1989), Ribeiro and Soumis (1994), Desaulniers et al. (1998) and Li (2013). Therefore, the SPP/SCP models are applied in this thesis to solve variants of vehicle and crew scheduling problems.

### 2.2 Shortest Path Problem with Resource Constraints

The shortest path with resource constraints (SPPRC) involves finding the minimum cost path between a source and a sink in a network while respecting all resource consumption constraints. Time, capacity and energy are some examples of resources. The SPPRC is an extension of the classical shortest path problem and the SPPRC is \( \mathcal{NP} \)-hard if there is at least one resource (Garey and Johnson, 1979).

Let \( G = (V, A) \) denote an acyclic directed network, where \( V \) is the set of vertices and \( A \) is the set of arcs. The source vertex of the network is denoted as \( o \) and the sink vertex is denoted as \( t \). A cost \( c_{ij} \) is associated with each arc \((i, j) \in A\). In this case of the SPPRC, only one resource is considered and \( w_{ij} \geq 0 \) denotes the resource consumption on arc \((i, j) \in A\). A path from \( o \) to \( t \) is feasible if the
2.2 Shortest Path Problem with Resource Constraints

The total consumption of the resource is less than or equal to \( W \). A binary decision variable \( x_{ij} \) is defined for each arc \((i, j) \in A\), where \( x_{ij} \) is equal to 1 if \((i, j)\) is part of the solution and 0 otherwise. The formulation of the SPPRC is as follows:

\[
\text{Minimize} \quad \sum_{(i,j) \in A} c_{ij} \cdot x_{ij} \tag{2.4}
\]

subject to,

\[
\sum_{i \in V} x_{oi} = 1 \quad \tag{2.5}
\]

\[
\sum_{i \in V} x_{it} = 1 \quad \tag{2.6}
\]

\[
\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = 0 \quad \forall i \in V \setminus \{o, t\} \quad \tag{2.7}
\]

\[
\sum_{(i,j) \in A} w_{ij} \cdot x_{ij} \leq W \quad \tag{2.8}
\]

\[
x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A \tag{2.9}
\]

Furthermore, in most cases of the SPPRC, each vertex in the network also has a certain cost and resource consumption. These characteristics of a vertex can easily be placed on its incoming arcs. For example, the resource consumption on arc \((i, j) \in A\) can be modified as \( w_{ij} = w_{ij} + w_j \), where \( w_j \) is the resource consumption on vertex \( j \in V \).

The SPPRC is commonly used to model the subproblem of a column generation method for solving various scheduling problems. The column generation method will be described in the next chapter. In this thesis, the SPPRC is applied to incorporate the resources related to vehicles and crews.
14

Combinatorial Optimization Problems
In this chapter, optimization algorithms that have been extensively applied in this thesis for solving combinatorial optimization problems are introduced and discussed. Sections 3.1 and 3.2 discuss exact solution approaches that guarantee to find the optimal solution for a given problem. Heuristics are typically applied for hard and large-scale problems to find near-optimal solutions in reasonable computation times. A metaheuristic provides a generic heuristic framework that can be adapted to solve a specific problem. In Section 3.3 a metaheuristic procedure is discussed.

3.1 Column Generation

A linear programming (LP) problem consists of a set of linear decision variables and is commonly solved by simplex algorithm. Column generation is a method that is used to solve LP problems involving a large number of variables. Ford and Fulkerson (1958) are the first to suggest the idea of implicitly handling all variables of a multi-commodity flow formulation within the framework of the simplex method. Inspired by this idea, Dantzig and Wolfe (1960) present a technique for the decomposition of linear programs that involves solving a sequence of reduced problems which contain only a small subset of variables. The first use of column generation was reported by Gilmore and Gomory (1961, 1963)
for solving the cutting stock problem. For more details on column generation, see Lübbecke and Desrosiers (2005).

Consider the following linear program with a large number of variables $|N|:

\[
\min \sum_{j \in N} c_j \cdot x_j
\]

subject to,

\[
\sum_{j \in N} a_{ij} \cdot x_j = b_i \quad \forall i \in M \tag{3.2}
\]

\[
x_j \geq 0 \quad \forall j \in N \tag{3.3}
\]

The formulation (3.1) - (3.3) is called the master problem (MP) and it maybe computationally intractable to explicitly handle all variables (or columns) in the problem. Therefore, in a column generation algorithm, the MP is initialized with only a small subset of columns $N' \subseteq N$. The reduced master problem is referred to as restricted master problem (RMP). On solving the RMP, the optimal values of dual variables corresponding to constraints (3.2) are obtained. The dual information is passed to a pricing problem, also called as subproblem, to generate new columns that have the potential of decreasing the objective value of the RMP. The pricing problem identifies a new set of columns (denoted as $N^P$) that have negative reduced cost, which are added to $N'$. The reduced cost of a column $j$ is calculated as shown in Equation (3.4), where $u_i$ is the value of dual variable corresponding to constraint $i \in M$. Column generation is an iterative framework between the RMP and the pricing problem, which terminates when there are no columns left with negative reduced costs. Furthermore, the optimal solution of the RMP is also the optimal solution of the MP. The pricing problem is an optimization problem and, in many applications, the problem is a shortest path problem or a shortest path problem with resource constraints.

\[
\tilde{c}_j = c_j - \sum_{i \in M} a_{ij} \cdot u_i \tag{3.4}
\]

Algorithm 1 gives an overview of the column generation algorithm. $Z_{LP}$ denotes the objective value of the RMP, $x$ and $u$ represent the optimal primal and dual solutions of the RMP, respectively.
Algorithm 1: Column Generation

1. **Initialization:** \( N' \leftarrow \text{InitialColumns}() \);
2. \( Z_{LP}, x, u \leftarrow \text{SolveRMP}(N') \);
3. \( N^P \leftarrow \text{SolvePricingproblem}(u) \);
4. if \( N^P \neq \emptyset \) then
   5. \( N' \leftarrow N' \cup N^P \);
6. go to 2;
7. end
8. return \( Z_{LP}, x \)

3.1.1 Integer Solutions

The column generation algorithm often terminates with an LP solution, where the variables take up fractional values. However, most problems are formulated as an integer program. For such cases, if the solution is integer at termination of the column generation procedure, then the solution is known to be the optimal solution. Otherwise, the column generation method is embedded in a branch-and-bound (B&B) framework to attain integer solutions. This procedure is known as the branch-and-price (B&P) method. For more information on B&P, see Barnhart et al. (1998).

Algorithm 2 gives an overview of the B&P method for a minimization problem. \( Z^* \) denotes the upper bound and initially, it is set to a sufficiently high value.

Let \( D \) be the set of unprocessed nodes in the B&B tree and it is initialized with the root node (denoted as \( d^0 \)). In each iteration of the B&B method, an unprocessed node \( d \in D \) is selected (Line 3). The selected node \( d \) is solved by column generation, which returns the optimal LP solution \( x^d \) of the reduced problem in \( d \) and its corresponding objective value \( Z^d_{LP} \). However, if there are no feasible solutions in the selected node, then the node is pruned by infeasibility (Line 6). If the LP objective value of the selected node is greater than or equal to the objective value of the current best solution (also called as *incumbent*), then the node is pruned by bound (Line 10). However, if the optimal LP solution \( x^d \) of the selected node is found to be integer, then the upper bound \( Z^* \) and the incumbent solution \( x^* \) are updated (Lines 14-15). Furthermore, the selected node is pruned by optimality (Line 16). A branching rule is incorporated in the B&B tree if the LP objective value is less than the upper bound. The branching operation splits the solution space of the selected node such that the current fractional solution \( x^d \) is excluded and integer solutions remain intact. A set of child nodes (denoted as \( D' \)) is created from the selected node and added to \( D \) (Lines 19-20). The algorithm terminates when there are no more nodes to process and returns the optimal integer solution.
**Algorithm 2: Branch and Price**

1. **Initialization:** $Z^* \leftarrow \infty, D \leftarrow \{d^0\};$
2. while $D \neq \emptyset$ do
   3. Select $d \in D;$
   4. $Z^d_{LP}, x^d \leftarrow \text{SolveNode}(d);$ // Solve by column generation
   5. if $x^d$ is empty then
      6. $D \leftarrow D \setminus \{d\};$ // Prune by infeasibility
      7. go to 2;
   8. end
   9. if $Z^d_{LP} \geq Z^*$ then
      10. $D \leftarrow D \setminus \{d\};$ // Prune by bound
      11. go to 2;
   12. end
   13. if $x^d$ is integer and $Z^d_{LP} < Z^*$ then
      14. $Z^* \leftarrow Z^d_{LP};$
      15. $x^* \leftarrow x^d;$
      16. $D \leftarrow D \setminus \{d\};$ // Prune by optimality
      17. go to 2;
   18. end
   19. $D' \leftarrow \text{Branching}(x^d);$  
   20. $D \leftarrow D \cup D';$
21. end
22. return $Z^*, x^*$

In most applications of column generation for solving routing and scheduling problems, the MP is formulated as an LP relaxation of the SPP given by Equations (2.1) - (2.3) in Chapter 2. A common branching strategy that has been applied for the SPP is the strategy suggested by Ryan and Foster (1981). In every fractional solution of the SPP, there must exist at least one pair of constraints $(r_1$ and $r_2 \in M)$ that has the following property:

$$0 < \sum_{j \in J(r_1, r_2)} x_j < 1 \quad (3.5)$$

$J(r_1, r_2)$ in Equation 3.5 represents the set of all variables that cover both constraints $r_1$ and $r_2$ simultaneously. By using this simple observation, Ryan and Foster (1981) impose an effective constraint branching technique that divides
3.2 Dynamic Programming

the solution space into the following two branches:

\[
\sum_{j \in J(r_1, r_2)} x_j \geq 1 \quad (1 - \text{branch}) \tag{3.6}
\]

\[
\sum_{j \in J(r_1, r_2)} x_j \leq 0 \quad (0 - \text{branch}) \tag{3.7}
\]

The 1 - branch implies that the pair of constraints must be covered simultaneously, while the 0 - branch implies that the pair of constraints must not be covered together. Furthermore, to implement the branches, relevant changes must be imposed in the RMPs and the pricing problems of the child nodes. For example, the 1 - branch could be implemented by adding a constraint in the RMP.

A complete enumeration of the B&B tree could be very time consuming. However, the B&P algorithm provides an effective framework for devising heuristic procedures to find feasible integer solutions in short computation times. For example, Barnhart et al. (1998) suggest the idea of choosing many pairs of constraints to eliminate the current fractional solution. A common heuristic strategy that is used for solving various scheduling problems is the variable fixing strategy, where all variables in the RMP that have fractional values above a certain value are fixed to 1 and the B&B tree is explored in a depth-first manner without backtracking (Desaulniers et al., 1998, Cordeau et al., 2001, Pepin et al., 2009 and Li, 2013).

3.2 Dynamic Programming

As mentioned earlier, in most applications, the pricing problem in the column generation method is formulated as a SPPRC that is described in Chapter 2. For more information on the SPPRC and its variants in a column generation setting, see Irnich and Desaulniers (2005).

Let us consider the SPPRC formulation given by Equations (2.4) - (2.9) in Chapter 2. \( G = (V, A) \) represents the acyclic directed network, where \( V \) is the set of vertices and \( A \) is the set of arcs. The source vertex is denoted as \( o \) and the sink vertex is denoted as \( t \). The cost and resource consumption on arc \( (i, j) \in A \) is represented as \( c_{ij} \) and \( w_{ij} \), respectively. The maximum resource consumption of a path from \( o \) to \( t \) is given by \( W \). The SPPRC has been extensively solved by dynamic programming (Irnich and Desaulniers, 2005). In a dynamic programming approach, more precisely a label-setting algorithm, partial paths from \( o \) are constructed in the form of a multi-dimensional resource vector called a label.
An initial label is associated with $o$ and labels are propagated forward through the network $G$ using resource extension functions (REFs) that are responsible for accumulating consumption of all resources along a path. Resource windows at each vertex in $G$ aids in discarding infeasible partial paths. Finally, a domination rule is applied to discard unpromising labels so that the enumeration of all feasible paths is avoided. The descriptions of the REFs, resource windows and domination rules for the SPPRC are as follows:

1. **Resource extension functions (REFs)**
   Given an arc $(i, j) \in A$ and a partial path $p$, the accumulated cost and resource consumption of $p$ at vertex $j \in V$ is calculated as follows:
   
   • Cost: $C^p_j = C^p_i + c_{ij}$.
   • Resource: $U^p_j = U^p_i + w_{ij}$

   , where $C^p_i$ and $U^p_i$ denote the accumulated cost and resource consumption of $p$ at vertex $i \in V$, respectively. Label of path $p$ at vertex $j$ is denoted as $l^p_j = (C^p_j, U^p_j)$. An initial label at source $o$ is defined as $l^o_p = (0, 0)$. Multiple labels are associated with each vertex and hence, $L_i$ denotes the set of all labels at vertex $i \in V$, where $L = \{l^1_1, l^2_1, ..., l^n_1\}$

2. **Resource windows**
   A label $l^p_i = (C^p_i, U^p_i)$ that represents a partial path $p$ from source $o$ to vertex $i$ is feasible if $U^p_i \in [0, W]$.

3. **Domination rules**
   Since multiple labels are associated with each vertex, a set of domination rules is applied to discard unpromising labels. Label $l^1_i$ at vertex $i$ dominates label $l^2_i$ if the conditions given by Equations (3.8) and (3.9) are satisfied. In such a case, $l^2_i$ can be discarded.
   
   $$C^1_i \leq C^2_i \quad (3.8)$$
   $$U^1_i \leq U^2_i \quad (3.9)$$

Algorithm 3 gives an overview of the generic labeling algorithm for solving the SPPRC. All the vertices in the network $G$ except $o$ are initialized with an empty set of labels. An initial label with zero resource consumption is assigned to $o$. Let $Q$ denote that set of vertices to be processed in the labeling algorithm, and $Q$ is initialized with $o$ (Line 1). In each iteration of the algorithm, a vertex $i$ is selected from $Q$ (Line 3). Each unprocessed label $l_i \in L_i$ is extended along each outgoing arc from $i$, which is represented as $(i, j)$, with the help of REFs(). For each extension, a new label $l^p_j$ is created at vertex $j$ (Line 6). Function CheckFeasibility() discards the label $l_j$ if it is infeasible (Line 7).
Algorithm 3: Labeling Algorithm

1. Initialization: $L_i \leftarrow \emptyset \quad \forall i \in V \setminus \{o\}$, $L_o \leftarrow \{l_o\}$, $Q \leftarrow \{o\}$;
2. while $Q \neq \emptyset$ do
   3. Select $i \in Q$;
   4. for $l_i \in L_i$ do
      5. for $(i, j) \in A$ do
         6. $l_j \leftarrow \text{REFs}(l_i, (i, j))$;
         7. CheckFeasibility($l_j$);
         8. if $l_j$ is feasible then
            9. CheckDomination($L_j, l_j$); // Discard dominated labels
            10. if $l_j$ is not dominated then
                11. $L_j \leftarrow L_j \cup \{l_j\}$;
                12. $Q \leftarrow Q \cup \{j\}$;
            13. end
         14. end
      15. end
   16. $Q \leftarrow Q \setminus \{i\}$
17. end
18. return $L_t$

feasible then it is compared against all the labels in $L_j$ with respect to the domination rules given by function CheckDomination(). All dominated labels at $j$ are discarded (Line 9). If $l_j$ is not dominated then it is added to $L_j$ and $j$ is added to $Q$ (Lines 10-13). The selected vertex $i$ is removed from $Q$ and the algorithm terminates when $Q$ is empty. The algorithm returns a set of all non-dominated labels at the sink $V$ vertex (denoted as $L_t$). The minimum cost feasible path from $o$ to $t$ can be filtered from $L_t$.

3.3 Adaptive Large Neighborhood Search

Large neighborhood search (LNS) heuristic was proposed by Shaw (1998) and is based on a local search framework such as simulated annealing (SA) or hill climber. The main principle of a local search algorithm is to improve an initial solution of an optimization problem by repeatedly applying a set of local changes. In LNS, a feasible solution is gradually improved by iteratively destroying and repairing the solution using a destroy and a repair method. The
Solution Methods

set of neighboring solutions of a given solution is termed a *neighborhood* and, in LNS, the destroy and repair methods implicitly define the neighborhood of the given solution. The *adaptive large neighborhood search* (ALNS) heuristic is an extension of the LNS, where multiple destroy and repair methods are defined to explore different neighborhoods within the same search. Furthermore, the ALNS framework contains an adaptive layer that controls the selection of each method based on its performance during the course of the search. The ALNS heuristic was proposed by Ropke and Pisinger (2006). For more information on the LNS and its extensions, see Pisinger and Ropke (2019).

Algorithm 4: Adaptive Large Neighborhood Search

1. **Initialization:** $s \leftarrow \text{InitialSolution}()$, $s^* \leftarrow s$;
2. $\rho \leftarrow \text{InitializeWeights}()$;
3. $\Omega \leftarrow \text{InitializeScores}()$;
4. $\nu \leftarrow \text{InitializeAttempts}()$;
5. **while** stop criteria not met **do**
6. Select destroy and repair methods $\mu \in \tau^-$ and $\gamma \in \tau^+$ using $\rho$;
7. $s' \leftarrow \text{Destroy}(s, \mu)$;
8. $s' \leftarrow \text{Repair}(s', \gamma)$;
9. **if** Accept($s, s'$) **then**
10. $s \leftarrow s'$;
11. **end**
12. **if** $f(s') < f(s^*)$ **then**
13. $s^* \leftarrow s'$;
14. **end**
15. $\Omega \leftarrow \text{UpdateScores}(\psi, \mu, \gamma)$;
16. $\nu \leftarrow \text{UpdateAttempts}(\mu, \gamma)$;
17. **if** update criteria met **then**
18. $\rho \leftarrow \text{UpdateWeights}(\Omega, \nu, \lambda)$;
19. $\Omega \leftarrow \text{ResetScores}()$;
20. $\nu \leftarrow \text{ResetAttempts}()$;
21. **end**
22. **end**
23. **return** $s^*$

Algorithm 4 gives an overview of the ALNS heuristic. Let $\tau^-$ denote the set of all destroy methods and $\tau^+$ denote the set of all repair methods in the ALNS framework. Each destroy method $\mu \in \tau^-$ is assigned a modifiable weight $\rho^\mu$. Similarly, each repair method $\gamma \in \tau^+$ is assigned a modifiable weight $\rho^\gamma$. All the destroy and repair methods have the same weight at the start of the heuristic. Let $s$ denote the current solution, $s'$ denote the neighboring solution and $s^*$ denote the best solution. An initial solution is computed, which serves as the input to the heuristic. In each iteration of the heuristic, a destroy method $\mu \in \tau^-$
and a repair method $\gamma \in \tau^+$ is selected (Line 6). The weights of the methods are used to select the methods using a roulette wheel principle. Well-performing methods have a higher probability of being selected. The probabilities of the destroy and repair methods are calculated as follows:

\begin{align*}
\zeta_\mu &= \frac{\rho^\mu}{\sum_{q \in \tau^-} \rho^q} \quad \forall \mu \in \tau^- \\
\zeta_\gamma &= \frac{\rho^\gamma}{\sum_{q \in \tau^+} \rho^q} \quad \forall \gamma \in \tau^+ 
\end{align*}

The selected destroy $\mu$ and repair $\gamma$ methods are applied to the current solution $s$ to obtain a new neighboring solution $s'$ (Lines 7-8). If $s'$ is accepted then it is set as the current solution $s$ (Lines 9-11). The SA acceptance criterion has been commonly used in the literature (see e.g. Ropke and Pisinger, 2006). Furthermore, if the objective value of $s'$ is better than the objective value of the incumbent solution $s^*$, then $s'$ is set as the new best solution (Lines 12-14). A score of $\psi$ is awarded to the selected destroy and repair methods based on the quality of the solution $s'$. The description of the score parameters is given in Table 3.1.

<table>
<thead>
<tr>
<th>Score ($\psi$)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>if the new solution is a new best solution</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>if the new solution is better than the current solution</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>if the new solution is accepted</td>
</tr>
<tr>
<td>$\psi_4$</td>
<td>if the new solution is rejected</td>
</tr>
</tbody>
</table>

Table 3.1: Score parameters for ALNS and $\psi_1 > \psi_2 > \psi_3 > \psi_4 \geq 0$.

$\Omega^\mu$ and $\Omega^\gamma$ denote the accumulated scores of destroy method $\mu$ and repair method $\gamma$, respectively. Similarly, $\nu^\mu$ and $\nu^\gamma$ denote the total number of times destroy method $\mu$ and repair method $\gamma$ have been selected during the course of the heuristic, respectively. They are updated at each iteration of the heuristic (Lines 15-16). In most cases, every time the heuristic performs certain number of iterations, the weights of the methods are updated based on the Equations (3.12) and (3.13), where $\lambda \in [0, 1]$ is known as the reaction factor that controls the degree of change in weights. Furthermore, $\Omega$ and $\nu$ for all destroy and repair methods are reset to 0 (Lines 17-21).

\begin{align*}
\rho^\mu &= (1 - \lambda) \cdot \rho^\mu + \lambda \cdot \frac{\Omega^\mu}{\nu^\mu} \quad \forall \mu \in \tau^- \\
\rho^\gamma &= (1 - \lambda) \cdot \rho^\gamma + \lambda \cdot \frac{\Omega^\gamma}{\nu^\gamma} \quad \forall \gamma \in \tau^+ 
\end{align*}
A maximum number of iterations or a time limit is commonly used as the termination criterion of the heuristic. The best solution $s^*$ is returned upon termination of the heuristic.
Chapter 4

Vehicle and Crew Scheduling: Models, Methods and Applications

In this chapter, a detailed literature review on models and solution methods for vehicle and crew scheduling problems that appear in public transport is given. The vehicle scheduling problem assigns buses to the timetabled trips such that every trip is covered by a bus. Subsequently, the crew scheduling problem is concerned with assigning the bus trips to drivers while satisfying numerous labor union regulations. Section 4.1 discusses the different extensions of the vehicle scheduling problem, which also includes scheduling of electric vehicles. In Section 4.2 the crew scheduling problem is discussed and an overview of the different labor regulations that were tackled in the literature is given. Section 4.3 gives an overview of the solution approaches proposed in the literature to integrate the vehicle and crew scheduling problems. Section 4.4 briefly discusses the recovery methods used to reduce impact of disruptions that occur during the day-of-operation. Finally, Section 4.5 addresses other future research directions.
4.1 Vehicle Scheduling Problem

4.1.1 Single Depot Vehicle Scheduling Problem

Given a bus depot, a set of timetabled trips with departure and arrival times and travel times between all pairs of bus stops, the objective of the single depot vehicle scheduling problem (SDVSP) is to find a minimum cost schedule such that each trip is assigned to a vehicle. A vehicle performs a feasible sequence of trips from the time it leaves the depot until it returns to the depot. The schedule of a vehicle is referred to as a block. Each block often starts with an empty move, i.e. a move without passengers, from the depot and ends with an empty move to the depot. Additionally, empty moves are placed between trips that do not end and start at the same bus stop. These empty moves are often referred to as deadheads. The cost of a block typically includes a fixed cost and a variable cost that is based on the total distance, in kilometers (km), covered by the vehicle during the day. The SDVSP is known to be solvable in polynomial time (Lenstra and Kan, 1981). It has been formulated as a linear assignment problem, a transportation problem, a minimum-cost flow problem, a quasi-assignment problem and a matching problem. For a detailed overview of models for the SDVSP, see Daduna and Paixão (1995) and Bunte and Kliewer (2009).

Quasi-assignment Model

In this section, the formulation of the SDVSP as a quasi-assignment problem (QAP) is described. Let $T$ be the set of timetabled trips. The vehicle scheduling network is defined as a directed graph $G = (V, A)$, where $V$ denotes the set of vertices and $A$ denotes the set of arcs. Each vertex $v \in V$ represents a trip and an arc $(i, j) \in A$ indicates that trip $j$ can be immediately covered by a vehicle after performing trip $i$. A deadhead is placed on the arc $(i, j)$ if the arrival bus stop of trip $i$ is not the same as the departure bus stop of trip $j$. Additionally, source $o \in V$ and sink $s \in V$ vertices are created that represent the depot. An arc from $o$ denotes the first pull-out deadhead from the depot and an arc to $s$ denotes the last pull-in deadhead to the depot of a vehicle. A path from $o$ to $s$ represents a block.

Let $c_{ij}$ be the cost of arc $(i, j) \in A$. The binary decision variable $y_{ij}$ indicates if a vehicle covers trip $j$ directly after trip $i$ or not. The QAP model of the SDVSP was given by Paixão and Branco (1987) and Freling et al. (2001) and the mathematical model is as follows:
4.1 Vehicle Scheduling Problem

\[
\text{Minimize } \sum_{(i,j) \in A} c_{ij} \cdot y_{ij} \quad (4.1)
\]

subject to,

\[
\sum_{j: (i,j) \in A} y_{ij} = 1 \quad \forall i \in T \quad (4.2)
\]

\[
\sum_{i: (i,j) \in A} y_{ij} = 1 \quad \forall j \in T \quad (4.3)
\]

\[
y_{ij} \in \{0, 1\} \quad \forall (i,j) \in A \quad (4.4)
\]

The objective of the SDVSP, given by (4.1), is to minimize the cost of vehicle schedule. Constraints (4.2) and (4.3) ensure that each trip is assigned to exactly one predecessor and one successor. These constraints define a totally unimodular restriction matrix and hence, the binary conditions on the variables are often relaxed to \( y_{ij} \geq 0 \) (Freling et al., 2001). An auction algorithm for the quasi-assignment problem is proposed by Freling et al. (2001). Their research was primarily motivated for their work on integration of vehicle and crew scheduling (Freling et al., 2003), where the SDVSP is solved many times to find a solution. The authors test the developed algorithm on instances from transport companies in the Netherlands (RET) and Portugal (CARRIS) that contain up to 1,328 timetabled trips. The algorithm is extremely fast and the instances could be solved in few seconds.

4.1.2 Multiple Depot Vehicle Scheduling Problem

The multiple depot vehicle scheduling problem (MDVSP) is an extension of the SDVSP, where multiple bus depots are present in the city network. A vehicle schedule must start and end at the same depot and the number of vehicles available at each depot is restricted. The MDVSP is known to be a \( \text{NP} \)-hard problem (Bertossi et al., 1987). Carpaneto et al. (1989) are the first authors to propose an exact method for the MDVSP. The authors describe a mixed integer programming (MIP) formulation based on an assignment formulation with additional path oriented flow conservation constraints and devise a branch-and-bound (B&B) algorithm to solve it. The literature on the MDVSP is abundant (see the surveys of Desrosiers et al., 1995, Desaulniers and Hickman, 2007 and Bunte and Kliewer, 2009). The MDVSP has commonly been formulated as a multi-commodity flow problem (MCF) or a set partitioning problem (SPP).
Multicommodity Flow Model

The MCF model of the MDVSP is described in Bodin et al. (1983) and Ribeiro and Soumis (1994). Let $K$ be the set of bus depots and $v_k$ be the maximum number of vehicles available at depot $k \in K$. $G^k = (V^k, A^k)$ denotes the vehicle scheduling network of depot $k \in K$ and $c^k_{ij}$ denotes the cost of arc $(i, j) \in G^k$. Decision variable $y^k_{ij}$ indicates if a vehicle from depot $k \in K$ covers trip $j$ immediately after trip $i$ or not. The MCF model is as follows:

\[
\text{Minimize} \sum_{k \in K} \sum_{(i,j) \in A^k} c^k_{ij} \cdot y^k_{ij} \tag{4.5}
\]

subject to,

\[
\sum_{k \in K} \sum_{j: (i,j) \in A^k} y^k_{ij} = 1 \quad \forall i \in T \tag{4.6}
\]

\[
\sum_{j: (j,i) \in A^k} y^k_{ji} - \sum_{j: (i,j) \in A^k} y^k_{ij} = 0 \quad \forall i \in V^k \setminus \{o^k, s^k\}, k \in K \tag{4.7}
\]

\[
\sum_{j: (o^k, j) \in A^k} y^k_{o^k,j} \leq v_k \quad \forall k \in K \tag{4.8}
\]

\[
y^k_{ij} \in \{0, 1\} \quad \forall (i, j) \in A, k \in K \tag{4.9}
\]

Constraints (4.6) ensure that each trip is covered exactly once. Flow conservation and capacity constraints are given by (4.7) and (4.8) respectively.

Forbes et al. (1994) solve the MCF model using a B&B algorithm. Löbel (1998) solve the linear programming (LP) relaxation of the MCF model by column generation method. The authors propose a new technique that is based on Lagrangian relaxations of the MCF model. The method is called Lagrangian pricing that generates arc variables for the master problem of the column generation method. Typically, the underlying network of the MCF model is a connection-based network, where the vertices in the network represent the trips and a pair of trips is connected by an arc if they are compatible with respect to time and space. However, Kliewer et al. (2006) propose a MCF model that is based on a time-space network structure. In the time-space network, each vertex corresponds to an arrival/departure time and arrival/departure bus stop of the trip. The network avoids the drawback of explicit consideration of all possible connections between compatible trips. Kliewer et al., 2006 apply an aggregation procedure for reducing the number of deadhead arcs without losing any feasible vehicle schedule. The authors used a commercial MIP solver to solve the resulting MCF model. Recently, Kulkarni et al. (2018) propose a new
4.1 Vehicle Scheduling Problem

MCF formulation, known as an inventory formulation, to model the MDVSP. In the inventory formulation network, only the arrival times and arrival locations of trips are denoted as vertices. Each compatible pair of trips is connected by a so-called inventory arc. The authors apply a column generation based heuristics to the inventory formulation.

Several heuristic solution approaches have been proposed in the OR literature to solve the MDVSP. One of the first heuristics to be successfully used in practice is the so-called concurrent scheduler that is proposed by Bodin et al. (1978). The concurrent scheduler is interpreted to be a “greedy” heuristic that considers trips in the increasing order of departure times and assigns a trip to an existing vehicle based on minimum deadhead time. If a feasible assignment of a trip to an existing vehicle does not exist, then a new vehicle is created and the trip is assigned to the new vehicle. Bertossi et al. (1987) propose a Lagrangian heuristic in which the trip covering constraints (4.6) are relaxed. Lamatsch (1992) develop a Lagrangian heuristic where the flow conservation constraints (4.7) are relaxed instead. Dell’Amico et al. (1993) propose a polynomial time heuristic algorithm that guarantees the use of minimum number of vehicles. The authors first solve a sequence of shortest path problems to build a good quality solution before applying different refinement procedures to improve the solution.

Set Partitioning Model

Ribeiro and Soumis (1994) formulated the MDVSP as a SPP with side constraints. A block is known to be the schedule of a vehicle and $B$ denotes the set of all feasible blocks. The cost of a block $b \in B$ is represented as $c_b$. Binary matrix $A^1$ is defined, where $a_{1t}^b$ is equal to 1 if block $b \in B$ covers trip $t \in T$ and 0 otherwise. Binary matrix $A^2$ is defined, where $a_{2k}^b$ is equal to 1 if block $b \in B$ belongs to depot $k \in K$ and 0 otherwise. Binary variable $y_b$ indicates if block $b \in B$ is selected as part of the schedule or not. This results in the following model:

$$\text{Minimize} \sum_{b \in B} c_b \cdot y_b$$  \hspace{1cm} (4.10)

subject to,

$$\sum_{b \in B} a_{1t}^b \cdot y_b = 1 \hspace{1cm} \forall t \in T \hspace{1cm} (4.11)$$

$$\sum_{b \in B} a_{2k}^b \cdot y_b \leq v_k \hspace{1cm} \forall k \in K \hspace{1cm} (4.12)$$

$$y_b \in \{0, 1\} \hspace{1cm} \forall b \in B \hspace{1cm} (4.13)$$
The objective function, given by (4.10), is to minimize the total cost. Set partitioning constraints (4.11) impose that each trip be covered by exactly one vehicle and constraints (4.12) ensure that the number of vehicles available per depot is restricted.

Ribeiro and Soumis (1994) propose column generation for solving the LP relaxation of the model (4.10) - (4.13), which is the restricted master problem (RMP). A subproblem is defined for every depot that is formulated as a shortest path problem and the authors solve it by dynamic programs. The authors propose a branch-and-price (B&P) method and use depth-first search as the branching strategy. Hadjar et al. (2006) propose a B&B algorithm that combines column generation, variable fixing and cutting planes. Oukil et al., 2007 present a stabilized column generation approach for the MDVSP, which efficiently handles highly degenerate instances. Pepin et al. (2009) compare the performance of five different heuristics for the MDVSP, namely, a truncated branch-and-cut method, a Lagrangian heuristic, a truncated column generation method, a large neighborhood search (LNS) heuristic and a tabu search heuristic. In the heuristic branch-and-cut method, a commercial MIP solver is used to solve the MCF formulation of the MDVSP and is terminated when an integer solution is found. The Lagrangian heuristic is similar to that of Lamatsch (1992). In the heuristic version of the column generation method, an early termination criterion is used where the column generation process is halted if the RMP objective value does not decrease much in the last few iterations. To find an integer solution, a depth-first branching strategy without backtracking is used and variables $y_b$ in the RMP that take up fractional values greater than or equal to 0.7 are rounded up to 1 at each node of the B&B tree. For the LNS heuristic, the authors propose to destroy at each iteration a part of the current solution and reoptimize using the column generation heuristic. Laurent and Hao (2009) propose an iterated local search (ILS) algorithm for the MDVSP.

Table 4.1 gives an overview of the literature on the MDVSP. The solution methods used for solving the MDVSP can be categorized into four methods: 1) mixed integer programming (MIP) methods that involve application of B&B methods or a commercial MIP solver such as CPLEX to obtain optimal solutions, 2) column generation (CG) approaches that are either based on exact B&P or heuristics, 3) metaheuristics (MH) and 4) heuristics (H) such as Lagrangian heuristics or a specialized heuristic procedure for the MDVSP. Löbel (1998) and Kliewer et al. (2006) succeeded in solving real-world instances from Germany to optimality involving up to 8,563 and 7,068 timetabled trips respectively. Pepin et al. (2009) tested the five heuristics on randomly generated instances with up to 1,500 trips and four depots. It was concluded that the column generation heuristic produces the best quality solutions when sufficient computational
4.1 Vehicle Scheduling Problem

time is available (maximum 2,300 seconds) and the LNS combined with column generation heuristic is the best alternative to obtain faster solutions without deteriorating too much solution quality.

<table>
<thead>
<tr>
<th>Authors</th>
<th>Model</th>
<th>Solution Method</th>
<th>Dataset</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dell’Amico et al. (1993)</td>
<td>MCF</td>
<td></td>
<td>10, 1,000</td>
<td>Random Lagrangian heuristics</td>
</tr>
<tr>
<td>Forbes et al. (1994)</td>
<td>MCF</td>
<td></td>
<td>3, 600</td>
<td>Random B&amp;B algorithm</td>
</tr>
<tr>
<td>Ribeiro and Soumis (1994)</td>
<td>SPP</td>
<td></td>
<td>6, 300</td>
<td>Random Lagrangian pricing</td>
</tr>
<tr>
<td>Libel (1998)</td>
<td>MCF</td>
<td></td>
<td>49, 24,906</td>
<td>Germany Time-space network</td>
</tr>
<tr>
<td>Kliwer et al. (2006)</td>
<td>MCF</td>
<td></td>
<td>5, 7,068</td>
<td>Germany ILS</td>
</tr>
<tr>
<td>Hadjar et al. (2006)</td>
<td>SPP</td>
<td></td>
<td>7, 2,100</td>
<td>Canada Lagrangian heuristics,</td>
</tr>
<tr>
<td>Pepin et al. (2009)</td>
<td>MCF, SPP</td>
<td></td>
<td>8, 1,500</td>
<td>Random TS, ILS</td>
</tr>
<tr>
<td>Laurent and Hao (2009)</td>
<td>MCF</td>
<td></td>
<td>8, 1,500</td>
<td>Random ILS</td>
</tr>
<tr>
<td>Kulkarni et al. (2018)</td>
<td>MCF</td>
<td></td>
<td>16, 3,000</td>
<td>Random Inventory formulation</td>
</tr>
</tbody>
</table>


One practical extension of the MDVSP is the multiple vehicle types vehicle scheduling problem (MVT-VSP). The MVT-VSP considers different vehicle types such as standard, double-decker and articulated buses that have different fixed and operational costs. Additionally, each vehicle type has a limited capacity at each depot. The problem is \( \mathcal{NP} \)-hard for the single depot case (Lenstra and Kan, 1981). Gintner et al. (2005) and Hassold and Ceder (2014) are some of the authors that have tackled the MVT-VSP. Another extension is the multiple depot vehicle scheduling problem with time windows (MDVSP-TW), where the departure times of each trip can be shifted within a prescribed time interval (see e.g. Desaulniers et al., 1998). A growing area of research is the integration of timetabling and vehicle scheduling that simultaneously minimizes travel times of passengers and operational cost of vehicles (see e.g. Ibarra-Rojas et al., 2014, Schmid and Ehmke, 2015, Fonseca et al., 2018 and Desfontaines and Desaulniers, 2018). Furthermore, Schöbel (2017) develop algorithms to integrate line planning, timetabling and vehicle scheduling.

4.1.3 Electric Vehicle Scheduling Problem

Electric buses are currently being introduced into the transportation system in many parts of the world. For instance, all city buses in Copenhagen will be
electric by 2025. The use of electric bus technology offers several challenges to the current planning process of transportation systems. Firstly, the charging infrastructure needs to be considered when operating electric buses. The different charging technologies are: 1) slow plug-in chargers installed at bus depots, 2) fast plug-in or pantograph chargers installed at terminals of bus lines or at bus stops, 3) overhead contact lines or inductive (wireless) chargers that are used to recharge buses during driving, and 4) battery swapping. See Häll et al. (2019) for a detailed description of the different charging technologies. The installation cost and the charging power in kilowatts (kW) of the different charging technologies are known to vary. Depot plug-in chargers have a low installation cost and a low charging power, whereas pantograph chargers have a high installation cost and a high charging power. The driving range in km of electric buses is dependent on the size of the battery in terms of kilowatt-hour (kWh). Electric buses with large battery packages are known to be more expensive, but have a longer driving range. Strategic decisions such as investment in charging infrastructure and the battery size of the bus fleet based on operational requirements (e.g. operating hours, frequency of lines) need to be made in order to minimize the total cost of ownership (TCO) (see e.g. Chen et al., 2018). Pelletier et al. (2019) present a electric bus fleet transition problem that determines bus replacement plans for transport companies to meet their electrification targets in a cost-effective way. Given different electric bus types and charger types, the problem considers investments decisions such as number of buses per bus type and number of chargers per charger type to purchase during the years 2020-2050. The battery capacity of buses varies from 110 to 650 kWh. A 110 kWh electric bus is estimated to have a driving range of 90 km and a 650 kWh electric bus has a driving range of approximately 370 km. Different charging infrastructures such as depot, fast, pantograph and inductive (wireless) charging are considered and their power vary from 50 to 300 kW. The authors assume that it takes approximately 13 hours to fully recharge a 650 kWh electric bus with a 50 kW depot charger. Another strategic planning problem is to find the cost-effective placement of chargers in the city bus network (Kunith et al., 2017 and Xylia et al., 2017).

At the tactical planning stage, the charging infrastructure is given and the charging schedule of the electric vehicles need to be determined. The electric vehicle scheduling problem (E-VSP), an extension of the VSP, is concerned with assigning a set of electric buses to a set of timetabled trips under the limited driving ranges and fixed charging locations. The objective of the E-VSP is to minimize the total operational cost that is contributed by fixed cost per vehicle and variable cost, which includes energy cost per km. A similar extension to the VSP is the vehicle scheduling problem with route constraints (VSP-RC), where a maximum route time constraint is present to ensure that the total time a vehicle is away from the depot is no more than a specified threshold (see e.g. Haghighi and Banihashemi, 2002 and Wang and Shen, 2007). Bodin et al. (1983)
show that any resource constrained VSP is \( \mathcal{NP} \)-hard. In the E-VSP, the power of the given charging infrastructure is often expressed in terms of the duration in minutes to fully recharge an electric vehicle. For example, Wen et al. (2016) assume that it takes two hours to fully recharge a vehicle and Li (2013) assume a battery service time of 10 minutes when the fast-charging or battery-charging technology is used. Similarly, the battery capacity of an electric bus has been expressed in terms of the maximum distance in kilometers it can cover without recharging.

Table 4.2 gives an overview of the electric bus technology and its related problems at different planning stages, namely strategic and tactical. The plug-in chargers, pantograph chargers and overhead contact lines are known to be the conductive chargers and the wireless charger is the inductive charger. Some authors that primarily tackle the tactical planning problem, utilize the charging stations located in the city network for recharging the vehicles and do not indicate the specific charging technology. It is assumed that a charging station has plug-in or pantograph chargers installed. Rogge et al. (2018) introduce the electric vehicle scheduling fleet size and mix problem with optimization of the charging infrastructure problem that tackles some of the strategic and tactical challenges. Given a set of timetabled trips and vehicle types, the problem determines the vehicle schedule to serve all trips and investment decisions such as the number of vehicles to buy per vehicle type. The charging infrastructure is considered to be installed at the depot and the problem also focuses on the number of chargers to buy per depot.

Li (2013) address the single-depot VSP for electric buses with battery swapping or fast charging at given battery stations. The authors present an arc formulation of the problem that consists of maximum distance before recharging or battery renewal constraints. The arc model is solved using a commercial MIP solver. The authors also reformulate the problem as a SPP or a path-based model. The SPP model is solved by a column generation method and a variable fixing strategy is used to find integer solutions. The authors assume that there exists one battery service station located at the depot and only a certain number of vehicles can be serviced at a time. Adler and Mirchandani (2017) present the alternative-fuel MDVSP, where other alternative-fuel vehicles such as hydrogen-gas vehicles and biofuel based vehicles that have limited driving ranges are considered. An exact B&P algorithm and a heuristic that is based on a concurrent scheduler algorithm (Bodin et al. (1978)) are proposed to solve the problem. Wen et al. (2016) address the E-VSP that allows vehicles to be recharged fully or partially at any of the given recharging stations. The recharging time is assumed to be a linear function and is proportional to the amount of battery charged. The authors propose an adaptive large neighborhood search (ALNS) heuristic for solving the E-VSP. Kooten Niekerk et al. (2017) state that the price of electricity significantly varies over the day and, in practice, the
### Table 4.2: Overview of the electric bus technology and its related problems at different planning stages.

<table>
<thead>
<tr>
<th>Authors</th>
<th>Planning stage</th>
<th>Problem</th>
<th>Charging infrastructure</th>
<th>Charging power (kW) or (minutes)</th>
<th>Battery capacity (kWh) or (km)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kunath et al. (2017)</td>
<td>Strategic</td>
<td>Cost-effective placement of charging infrastructure and battery capacity of buses</td>
<td>Fast-charging</td>
<td>300 kW</td>
<td>60 - 150 kWh</td>
</tr>
<tr>
<td>Xylia et al. (2017)</td>
<td>Strategic</td>
<td>Locating charging infrastructure</td>
<td>Conductive and inductive chargers</td>
<td>200-300 kW</td>
<td>60 kWh</td>
</tr>
<tr>
<td>Chen et al. (2018)</td>
<td>Strategic</td>
<td>Cost analysis of charging infrastructure</td>
<td>Charging stations, inductive chargers and battery swapping</td>
<td>80-120 kW</td>
<td>324 kWh</td>
</tr>
<tr>
<td>Pelletier et al. (2019)</td>
<td>Strategic</td>
<td>Electric bus fleet transition problem</td>
<td>Depot plug-in, fast plug-in pantograph and wireless chargers</td>
<td>50-300 kW</td>
<td>110-650 kWh</td>
</tr>
<tr>
<td>Rogge et al. (2018)</td>
<td>Strategic, Tactical</td>
<td>Electric bus fleet size and mix problem with optimization of charging infrastructure</td>
<td>Depot charging</td>
<td>48-60 kW</td>
<td>90-380 kWh</td>
</tr>
<tr>
<td>Li (2013)</td>
<td>Tactical</td>
<td>Electric vehicle scheduling</td>
<td>Fast-charging and battery swapping</td>
<td>10 minutes</td>
<td>120-150 km</td>
</tr>
<tr>
<td>Adler and Mirchandani (2017)</td>
<td>Tactical</td>
<td>Alternative-fuel vehicle scheduling</td>
<td>Fuelling or charging stations</td>
<td>10 minutes</td>
<td>120 km</td>
</tr>
<tr>
<td>Wen et al. (2016)</td>
<td>Tactical</td>
<td>Electric vehicle scheduling</td>
<td>Charging stations</td>
<td>120 minutes</td>
<td>150 km</td>
</tr>
<tr>
<td>Kooten Nierker et al (2017)</td>
<td>Tactical</td>
<td>Electric vehicle scheduling</td>
<td>Charging stations</td>
<td>120 kW</td>
<td>122-144 kWh</td>
</tr>
</tbody>
</table>

Cost is dependent on the time when the electricity is taken from the grid. The charging stations are most likely to be at the depots or terminals of lines and each charging station has a certain space capacity that determines the number of vehicles that can be charged simultaneously. The charging station also has an energy capacity, and larger capacities imply that the electric vehicles can be charged faster. The authors propose two models to solve the E-VSP. The first model is a MIP model with continuous variables for battery charge. For every trip, an extra variable is assigned that keeps track of the charge at the start of a trip. The model considers only linear charging behaviour of the batteries and a constant price of electricity during the day. The second model allows for non-linear charging behaviour of the batteries and takes the actual electricity prices during the day into account. The second model is also reformulated as a SPP so that it can be solved by a column generation method. The authors describe three solution methods for the second model; namely, a MIP solver and column generation heuristics that are based on linear programming and Lagrangian relaxations.
Table 4.3 gives an overview of the literature on the E-VSP. Li (2013) compares the performance of an arc model that is solved by a MIP solver and an heuristic column generation method. For the large instances with 947 timetabled trips, the LP relaxation of the arc model is not solved to optimality by the MIP solver in 12 hours. The column generation based method provided solutions that have an average optimality gap of 7%, and the average computation time was 72 hours. Adler and Mirchandani (2017) test an exact B&P algorithm only on subsets of the original data, which contained 4,373 timetabled trips. The subsets of the data had up to 72 trips, eight refuelling stations and four depots. The B&P algorithm took between two and 12 hours of computation time to solve the small instances. In comparison, the heuristic that is based on the concurrent scheduler algorithm took less than a second, but the average optimality gap was 11.80%. Kooten Niekerk et al. (2017) use MIP models to solve only the small instances that had up to 241 timetabled trips, and column generation based methods are used to solve the larger instances.

<table>
<thead>
<tr>
<th>Authors</th>
<th>Solution Method</th>
<th>Dataset</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MIP  CG  MH  H</td>
<td>K</td>
<td>T</td>
</tr>
<tr>
<td>Li (2013)</td>
<td>●  ●</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Adler and Mirchandani (2017)</td>
<td>●  ●</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Wen et al. (2016)</td>
<td>●  ●</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>Kooten Niekerk et al. (2017)</td>
<td>●  ●</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.3: Overview of the literature on the E-VSP. Solution method: MIP-mixed integer programming methods, CG-column generation (exact branch-and-price or heuristics), MH-metahuristics and H-heuristics. Dataset: |K|-number of depots, |T|-number of timetabled trips and Test-random or real-world instances. Other abbreviations: ALNS-adaptive large neighborhood search.

In summary, the scheduling of electric vehicles is a rapidly growing area of research primarily due to the technological challenges it presents. The electric technology enforces adjustments to the current transportation planning process in order to efficiently operate the vehicles. There has been an immense increase in the literature for tackling strategic problems such as the investments in the charging infrastructure, the electric vehicles and the placement of charging stations in the city network. Moreover, the E-VSP has been proposed to find cost-efficient vehicle schedules for a given charging infrastructure. However, to the best of our knowledge, there has not been any literature that simultaneously handles timetabling and the E-VSP such that the charging activities for the electric vehicles are incorporated into the timetable. This point has also been briefly discussed in Häll et al. (2019); however, a solution approach is not proposed. Integration of timetabling and electric vehicle scheduling would be an interesting topic of research that could potentially provide more insights into efficiently operating a transportation system with electric vehicles.
4.2 Crew Scheduling Problem

The crew scheduling problem (CSP) is also referred to as the driver scheduling problem (DSP) in the literature to indicate the scheduling problem in the bus industry. However, the CSP that appears in the bus industry is similar to the problems that arise in other industries such as airline and railway. For a detailed literature survey on the railway crew scheduling problem, see Heil et al. (2019). The CSP is the second step in the planning process of operations for a public transport company, where a set of bus trips is given. The set of bus trips or tasks includes the set of timetabled trips and deadheads performed by the vehicles. The schedule of a driver for a day is known as a duty. The feasibility of a duty is influenced by various labor rules and regulations that govern the working conditions of the drivers. A cost is associated with each duty and, in most cases, the wages paid to the driver are used as an estimate for the cost. Given a set of bus trips, the CSP is concerned with finding an optimal set of duties that covers all bus trips with minimal cost and satisfies all the labor regulations.

4.2.1 Labor Regulations

A public transport company is often associated with different labor unions that impose varying rules and regulations. Most regulations are concerned with the working period of the drivers and ensure that the drivers receive a sufficient number of breaks. The following are the most common regulations that are found in the literature and apply to most transport companies:

- **Maximum duration**
  Duration of a duty is defined as the period of time between the start and end of a driver's duty. The duration of a driver's duty can never exceed a certain limit (see e.g. Desrochers and Soumis, 1989, Fores et al., 2002 and Yunes et al., 2005). In most cases, there are also limitations on the total working period of the drivers that includes only the total driving time and the break periods are not included. For example, Fores et al. (2002) considers the maximum duration of a duty to be 12 hours and the maximum working duration to be nine hours.

- **Minimum break duration**
  This specifies that breaks must be at least a certain duration. Fores et al. (2002) and Yunes et al. (2005) set the minimum break duration as 30 minutes. In some cases, there are also restrictions on the maximum duration of the break. For example, Chen (2013) considers the maximum duration of a break to be 150 minutes.
• **Maximum time without break**
The duration between breaks cannot exceed a certain limit. This rule ensures that a driver does not drive a bus for a prolonged period without a break. Fores et al. (2002) considers the maximum duration without a break to be five hours.

• **Maximum number of pieces of work**
A driver may cover only a few consecutive bus trips before the driver takes a break or is relieved of duty. A *piece of work* is a feasible sequence of consecutive trips of a single bus that can be covered by a driver (Freling et al., 2003 and Kliwer et al., 2012). A duty is typically composed of pieces of work that are separated by breaks. In practice, a duty consists of two-three pieces of work. Desrochers and Soumis (1989) set the maximum number of pieces of work as three and Freling et al. (2003) considers a maximum of two pieces of work. In most cases, the maximum duration of a piece of work is regarded as the maximum time without break. For example, Desrochers and Soumis (1989) consider the maximum duration of a piece of work to be six hours and Chen (2013) considers it to be four hours. A duty with two pieces of work essentially denotes one bus or vehicle change for the driver. Some authors (see e.g. Portugal et al., 2009) have also used a rule to restrict the number of vehicle changes that a driver can make during his/her duty.

• **Multiple duty types**
A duty can be categorized into one of several types depending on its characteristics. The most common categorization is based on the starting times of the duties during the day such as early or late duty (see e.g. Freling et al., 2003 and Li et al., 2015). Therefore, the CSP considers multiple duty types and the rules often vary for each duty type.

• **Multiple driver depots**
Some authors (see e.g. Boschetti et al., 2004) have considered multiple driver depots, where drivers are allowed to start their respective duties from any of the given depots. However, a driver is required to end his/her duty at the same depot where the duty was started.

In some cases, some of the above rules such as the maximum duration and maximum time without break have been considered as a “soft” rule that is allowed to be violated (see e.g. Desrochers and Soumis, 1989 and Portugal et al., 2009). However, an additional cost, such as an overtime rate, is usually included to deter the violating rules. The various break rules enforce drivers to travel between bus stops and depots in the city network to take a break or start/end their respective duties. The travel or deadheading activities of the drivers are essential for creating feasible duties. Some examples include travel by
foot (Wren et al., 2003), taxis (Abbink et al., 2011 and Potthoff et al., 2010) and as passengers, where the driver is a passenger on another bus (Fores et al., 2002 and Wren et al., 2003). However, in most case, only the travel times between specific locations are taken into account for scheduling the drivers and the mode of transport is not considered (see e.g. Smith and Wren, 1988, Desrochers and Soumis, 1989 and Boschetti et al., 2004).

4.2.2 Mathematical Model

Common formulations of the CSP are based on the SPP or the set covering problem (SCP), where the formulation is used as a duty selection module with the selected duties covering all bus trips at minimum cost (Ibarra-Rojas et al., 2015). Let \( S \) be the set of all bus trips that includes the timetabled trips and the deadheads performed by the vehicles. Let \( D \) be the set of all feasible duties. The cost of a duty \( d \in D \) is represented as \( c_d \). Binary matrix \( A^3 \) is defined, where \( a_{sd}^3 \) is equal to 1 if duty \( d \in D \) covers bus trip \( s \in S \) and is 0 otherwise. Binary variables \( x_d \) indicate if duty \( d \in D \) is selected as part of the schedule or not. The SPP model of the CSP is given as,

\[
\text{Minimize } \sum_{d \in D} c_d \cdot x_d \quad (4.14)
\]

subject to,

\[
\sum_{d \in D} a_{sd}^3 \cdot x_d = 1 \quad \forall s \in S \quad (4.15)
\]

\[
x_d \in \{0, 1\} \quad \forall d \in D \quad (4.16)
\]

The objective of the CSP, given by (4.14), is to minimize the total cost of duties. Constraints (4.15) ensure that each bus trip is covered by exactly one duty. In a SCP model, the equality signs in (4.15) are replaced by “\( \geq \)” signs, which indicate that drivers are allowed to travel in the city network as a passenger on the bus. In most cases, there are additional side constraints such as the allowed number of duties per duty type (see e.g. Desrochers and Soumis, 1989) and the number of duties per depot (see e.g. Boschetti et al., 2004). Lourenço et al. (2001) consider multiple objective functions to the problem such as minimizing number of duties and minimizing number of duties with only one piece of work. The authors also consider minimizing the number of vehicle changes since a change of the driver responsible for a vehicle can cause disruptions to a company’s operation.
4.2 Crew Scheduling Problem

4.2.3 Solution Methods

To find the optimal solution, all the feasible duties have to be considered in the SCP formulation. However, the formulation is intractable by exhaustive enumeration techniques because of the large number of feasible duties for all practical problems. Some authors, e.g. Smith and Wren (1988) and Wren et al. (2003), heuristically generate a feasible subset of duties for the SCP and solve it by a specialized B&B algorithm or a commercial MIP solver. Wren et al. (2003) focus more on combining theory and practice to create a user-friendly and flexible system. Similarly, Portugal et al. (2009) present SPP based models that are solved by a MIP solver and aim to implement the models as part of a planning system. Some authors consider solving the SCP by metaheuristic procedures such as genetic algorithms (GA) and tabu search (Lourenço et al., 2001 and Li and Kwan, 2003). Caprara et al. (1999) present a Lagrangian-based heuristic for solving the SCP. The algorithm was designed to solve large-scale crew SCP instances arising from crew scheduling in an Italian railway company.

Desrochers and Soumis (1989) are the first authors to propose a column generation approach, more precisely a B&P method, for the CSP. The subproblem of the column generation method is modelled as a shortest path problem with resource constraints (SPPRC) that is solved by a dynamic programming approach. The authors use a branching rule that is similar to the one developed by Ryan and Foster (1981).

Solving large instances of the CSP by column generation approaches have been reported in the literature as being computationally expensive due to the need to solve SPPRC at every iteration (Wren et al. (2003), Yunes et al. (2005) and Ibarra-Rojas et al. (2015)). Yunes et al. (2005) compare a constraint programming (CP) approach and a dynamic programming technique, which was suggested by Desrochers and Soumis (1989), to solve the SPPRC. It was reported that solving the SPPRC by CP in the column generation setting outperforms that of the dynamic programming in terms of computation time. Steinzen (2007) and Kliwer et al. (2012) tackle the CSP as part of the integrated VSP and CSP (VCSP) and propose acceleration and heuristic procedures for solving the SPPRC by dynamic programming. Freling et al. (2003) and Huisman et al. (2005) use the pieces of work to model the subproblem of the CSP for solving the VCSP. A piece of work can be viewed as a partial duty and the authors enumerate all pieces of work for a given vehicle schedule.

Fores et al. (2002) do not solve the SPRRC to generate duties, but create a pregenerated duty set with a large number of feasible duties (around 1.5 million duties for the largest instance with 1,500 bus trips) and then apply column generation to evaluate the pregenerated feasible duties at each iteration. Chen
proposed a similar approach, but solve the SPPRC when there are no more duties in the pregenerated duty set that will improve the solution objective. Several metaheuristic approaches have been proposed in the literature to solve the subproblem; some examples include GA (Mauri and Lorena, 2007 and Santos and Mateus, 2009) and hyper-heuristics (Li et al., 2015). Santos and Mateus (2009) initially use the genetic algorithm to solve the SPPRC and later use an exact method to solve the SPPRC for ensuring optimality. Li et al. (2015) generate all feasible variables for a given instance and several heuristics (local search, swap heuristic and greedy based heuristic) are devised to select a subset of feasible variables at each iteration of the column generation framework. For an instance with 500 trips, the number of feasible duties was found to be around 8.3 million. Kecskeméti and Bîlăci (2013) present a hybrid of column generation and evolutionary algorithm (EA). The EA is used to solve large-scale set covering problems and it is re-run after a number of new columns are generated to find a feasible solution quickly. Column generation has been used in combination with Lagrangian relaxation for solving the CSP in the railway industry (e.g. Abbink et al., 2005 and Abbink et al., 2011) and for solving the CSP in the context of the VCSP (see e.g. Freling et al. (2003)), which will be briefly described in Section 4.3.

Table 4.4 gives an overview of the literature on the CSP. The solution methods used for solving the CSP can be categorized into three methods: 1) column generation (CG) approaches that are either based on exact B&P or heuristics, 2) metaheuristics (MH) that are primarily devised to solve large-scale SCP and 3) heuristics (H) that focus on generating a large subset of feasible duties for a SCP model, which is then solved by a commercial MIP solver. In conclusion, most of the research carried out on the CSP has been motivated by real-world applications where transport companies have to handle many labor regulations. Furthermore, most of the solution methods proposed in literature have been developed to be part of commercial decision support tools (see e.g. Desrochers and Soumis, 1989, Lourenço et al., 2001, Fores et al., 2002, Wren et al., 2003 and Portugal et al., 2009).

### 4.3 Integrated Vehicle and Crew Scheduling Problem

Given a set of timetabled trips, the integrated vehicle and crew scheduling problem (VCSP) aims to find a minimum cost schedule for the vehicles and the crews such that both the vehicle and crew schedules are feasible and mutually compatible. In addition to assigning the timetabled trips to a vehicle and a driver, any deadheads in the vehicle schedule need to be assigned a driver. Furthermore,
4.3 Integrated Vehicle and Crew Scheduling Problem

<table>
<thead>
<tr>
<th>Authors</th>
<th>Solution Method</th>
<th>Dataset</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith and Wren (1988)</td>
<td>CG</td>
<td>309</td>
<td>UK</td>
</tr>
<tr>
<td>Desrochers and Soumis (1989)</td>
<td>MH</td>
<td>235</td>
<td>USA and UK</td>
</tr>
<tr>
<td>Lourenço et al. (2001)</td>
<td>H</td>
<td>348</td>
<td>Portugal</td>
</tr>
<tr>
<td>Fores et al. (2002)</td>
<td>MG H</td>
<td>1,500</td>
<td>UK</td>
</tr>
<tr>
<td>Wren et al. (2003)</td>
<td>MG H</td>
<td>1,274</td>
<td>UK</td>
</tr>
<tr>
<td>Li and Kwan (2003)</td>
<td>MG H</td>
<td>1,873</td>
<td>UK</td>
</tr>
<tr>
<td>Li and Kwan (2003)</td>
<td>MG H</td>
<td>1,873</td>
<td>GA</td>
</tr>
<tr>
<td>Li and Kwan (2003)</td>
<td>MG H</td>
<td>1,873</td>
<td>GA</td>
</tr>
<tr>
<td>Li et al. (2015)</td>
<td>MG H</td>
<td>265</td>
<td>China</td>
</tr>
</tbody>
</table>

Table 4.4: Overview of literature on the CSP. **Solution method:** CG-column generation (exact branch-and-price or heuristics), MH-metaheuristics and H-heuristics. **Dataset:** |S| - number of bus trips and Test-random or real-world instances. Other abbreviations: B&B-branch-and-bound, TS-tabu search, GA-genetic algorithm, CP-constrained programming and EA-evolutionary algorithm.

in most cases, a **continuous attendance** is required, i.e. there is always a driver present when the vehicle is outside the depot (Freling et al., 2003 and Huisman et al., 2005). The solution methods in the literature for tackling the VCSP fall into one of the following three categories (Freling et al. (2003)):

1. Inclusion of vehicle considerations in the crew scheduling problem and the vehicle scheduling is carried out afterwards (crew first - vehicle second).

2. Inclusion of crew considerations in the vehicle scheduling problem and the crew scheduling is carried out afterwards (vehicle first - crew second).

3. Complete integration of vehicle and crew scheduling

The first and second categories are recognized as partial integration methods of the vehicle and crew scheduling problems. For a review on the partial integration methods, see Freling et al. (2003). One interesting approach that belongs to neither of the partial integration categories is proposed by Gintner et al. (2008). The authors consider a set of optimal vehicle schedules to find the best crew schedule.
4.3.1 Mathematical Model

In this section, the mathematical model presented by Friberg and Haase (1999) for the complete integration of vehicle and crew scheduling is described. The authors present a formulation of the VCSP that combines the approaches of Desrochers and Soumis (1989) and Ribeiro and Soumis (1994) for solving the CSP and VSP respectively. The VCSP is formulated as a SPP with additional constraints that link the crew and vehicle schedules. A linking constraint ensures that a deadhead is covered by a crew member only if it is covered by a vehicle. Table 4.5 gives the descriptions of the notations in the VCSP.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>Set of timetabled trips.</td>
</tr>
<tr>
<td>$F$</td>
<td>Set of all deadheads.</td>
</tr>
<tr>
<td>$B$</td>
<td>Set of all blocks.</td>
</tr>
<tr>
<td>$D$</td>
<td>Set of all feasible duties.</td>
</tr>
<tr>
<td>$c_1^b$</td>
<td>Cost of block $b \in B$.</td>
</tr>
<tr>
<td>$c_2^d$</td>
<td>Cost of duty $d \in D$.</td>
</tr>
<tr>
<td>$A_1$</td>
<td>Binary matrix, where $a_{1b}^t$ is 1 if block $b \in B$ covers trip $t \in T$ and is 0 otherwise.</td>
</tr>
<tr>
<td>$A_4$</td>
<td>Binary matrix, where $a_{4d}^t$ is 1 if duty $d \in D$ covers trip $t \in T$ and is 0 otherwise.</td>
</tr>
<tr>
<td>$A_5$</td>
<td>Binary matrix, where $a_{5b}^f$ is 1 if block $b \in B$ contains deadhead $f \in F$ and is 0 otherwise.</td>
</tr>
<tr>
<td>$A_6$</td>
<td>Binary matrix, where $a_{6d}^f$ is 1 if duty $d \in D$ contains deadhead $f \in F$ and is 0 otherwise.</td>
</tr>
<tr>
<td>$y_b$</td>
<td>A binary variable that indicates if block $b \in B$ is part of the schedule or not.</td>
</tr>
<tr>
<td>$x_d$</td>
<td>A binary variable that indicates if duty $d \in D$ is selected as part of the schedule or not.</td>
</tr>
</tbody>
</table>

Table 4.5: Descriptions of the notations in the VCSP.

The mathematical model for the VCSP is as follows:

$$\text{Minimize } \sum_{b \in B} c_1^b \cdot y_b + \sum_{d \in D} c_2^d \cdot x_d \quad (4.17)$$

subject to,

$$\sum_{b \in B} a_{1b}^t \cdot y_b = 1 \quad \forall t \in T \quad (4.18)$$

$$\sum_{d \in D} a_{4d}^t \cdot x_d = 1 \quad \forall t \in T \quad (4.19)$$

$$\sum_{d \in D} a_{6d}^f \cdot x_d - \sum_{b \in B} a_{5b}^f \cdot y_b = 0 \quad \forall f \in F \quad (4.20)$$

$$y_b \in \{0, 1\} \quad \forall b \in B \quad (4.21)$$

$$x_d \in \{0, 1\} \quad \forall d \in D \quad (4.22)$$
The objective of the VCSP, given by (4.17), is to minimize the total cost of blocks and duties. Constraints (4.18) and (4.19) ensure that each timetabled trip is assigned to a vehicle and a driver respectively. Constraints (4.20) are the deadhead linking constraints.

Haase et al. (2001) present another SPP model with side constraints that only involves crew variables for the single depot VCSP. Inclusion of vehicle cost and the side constraints in the formulation ensure that an overall optimal solution is found after deriving a compatible vehicle schedule. Freling et al. (2003) present a mathematical formulation for the single-depot VCSP that is a combination of the QAP formulation for the VSP and the SPP/SCP formulation for the CSP. For the multiple depot VCSP, the mathematical formulation presented in the literature involves the MCF formulation for the VSP and the SPP/SCP formulation for the CSP (Huisman et al., 2005, Borndörfer et al., 2008, Mesquita and Paias, 2008 and Steinzen et al., 2010).

4.3.2 Solution Methods

Friberg and Haase (1999) propose the first exact algorithm for the single depot case of the VCSP that uses the mathematical model (4.17)-(4.22). A B&P method for obtaining optimal solutions is proposed and a column generation procedure is performed to generate both vehicle and crew variables. Haase et al. (2001) also propose an exact B&P approach for solving the single depot case of the VCSP. The authors utilize several acceleration strategies such as omission of redundant constraints in the master problem, dynamic generation of bus count constraints and substitution of partitioning constraints, to speed up the solution process. Freling et al. (2003) propose a column generation procedure based on Lagrangian relaxation of the single depot VCSP model, where the VSP is formulated as the QAP and the CSP is formulated as the SPP/SCP. Column generation is generally applied in the context of LP; however, the authors state that the number of constraints in the VCSP model make the LP relaxation an unrealistic option. The authors relax all the constraints related to the crew in a Lagrangian way. The column generation procedure only involves generating crew variables and the Lagrangian subproblem involving the single depot VSP is solved using the auction algorithm (Freling et al., 2001). The authors use subgradient optimization to solve the Lagrangian dual problem approximately. Furthermore, the columns that are generated to compute the lower bound are used to construct a feasible solution either by applying the heuristics of Caprara et al. (1999) or by using a commercial MIP solver.
Gaffi and Nonato (1999) introduce the multiple depot vehicle and crew scheduling problem (MD-VCSP). The mathematical formulation is similar to Freling et al. (2003) and the authors propose a solution approach that is based on Lagrangian relaxation with column generation. The authors assume that a driver is assigned to one vehicle for the whole planning period and all pieces of work start and end at a depot. This assumption is considered for a particular application, and is does not hold in general for most applications. It has been argued in the literature that this assumption simplifies the problem and makes it computationally more tractable (Huisman et al., 2005 and Steinzen et al., 2010). Huisman et al. (2005) extend the work carried out by Freling et al. (2003) and the solution approach is applied to the MD-VCSP. Borndörfer et al. (2008) propose a similar method to that of Freling et al. (2003) and Huisman et al. (2005) to integrate the vehicle and driver scheduling problems. However, the authors use bundle techniques for the solution of the Lagrangian relaxations. The authors state that the advantages of the bundle method are that it provides high quality bounds and automatically generates primal information. Steinzen et al. (2010) also use column generation in combination with Lagrangian relaxation. However, the authors use a time-space network to represent the underlying network of the vehicle scheduling problem. Mesquita and Paia (2008) solve the LP relaxation of the MD-VCSP model using column generation. If the resulting solution is not integer, then the authors use a B&B procedure over the set of feasible crew duties that was generated while solving the LP relaxation to obtain a feasible integer solution. Mesquita et al. (2009) is an extension of the work carried out in Mesquita and Paias (2008), where the authors propose different branching strategies for a B&P method.

Groot and Huisman (2008) apply the same procedure as Huisman et al. (2005) for solving the MD-VCSP. However, the authors discuss several approaches for splitting large instances of MD-VCSP into smaller ones in order to apply an integrated approach within a reasonable computation time without significantly deteriorating the quality of the solutions. The splitting procedure involves assigning each trip to a depot. Laurent and Hao (2008) present a constraint programming based model for the VCSP and propose a greedy randomized adaptive search procedure (GRASP) for solving it. Similarly, De Leone et al. (2011) propose a GRASP algorithm for solving the VCSP. Table 4.6 gives an overview of the literature on the VCSP. The solution methods used for solving the VCSP can be categorized into two methods: 1) column generation (CG) approaches that are either based on LP relaxation or Lagrangian relaxation and 2) metaheuristics (MH). Some authors such as Friberg and Haase (1999) and Haase et al. (2001) have proposed exact B&P methods; however, almost all column generation approaches are based on heuristics for solving large instances of VCSP.

Freling et al. (2003) are the first authors to make a comparison between the
4.3 Integrated Vehicle and Crew Scheduling Problem

<table>
<thead>
<tr>
<th>Authors</th>
<th>Model</th>
<th>Solution Method</th>
<th>Dataset</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friberg and Haase (1999)</td>
<td>SPP SPP</td>
<td>CG MH</td>
<td>1 30 Random B&amp;P</td>
<td></td>
</tr>
<tr>
<td>Haase et al. (2001)</td>
<td>SPP</td>
<td>CG MH</td>
<td>1 250 Random B&amp;P</td>
<td></td>
</tr>
<tr>
<td>Fredling et al. (2003)</td>
<td>QAP SPP/SCP</td>
<td>CG MH</td>
<td>1 238 Netherlands Lagrangian relaxation</td>
<td></td>
</tr>
<tr>
<td>Huisman et al. (2005)</td>
<td>MCF SPP/SCP</td>
<td>CG MH</td>
<td>4 653 Netherlands Lagrangian relaxation</td>
<td></td>
</tr>
<tr>
<td>Borndörfer et al. (2008)</td>
<td>MCF SPP</td>
<td>CG MH</td>
<td>3 1,414 Germany Lagrangian relaxation</td>
<td></td>
</tr>
<tr>
<td>Laurent and Hau (2008)</td>
<td>Constraint programming</td>
<td>CG MH</td>
<td>1 249 Real-world GRASP</td>
<td></td>
</tr>
<tr>
<td>Mesquita and Paias (2008)</td>
<td>MCF SPP/SCP</td>
<td>CG MH</td>
<td>4 400 Random LP relaxation</td>
<td></td>
</tr>
<tr>
<td>Mesquita et al. (2009)</td>
<td>MCF SPP/SCP</td>
<td>CG MH</td>
<td>4 238 Portugal B&amp;P</td>
<td></td>
</tr>
<tr>
<td>Steinzen et al. (2010)</td>
<td>MCF SPP</td>
<td>CG MH</td>
<td>4 640 Random Time-space network</td>
<td></td>
</tr>
<tr>
<td>De Leone et al. (2011)</td>
<td>MCF SPP</td>
<td>CG MH</td>
<td>2 400 Random GRASP</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.6: Overview of literature on the VCSP. **Model:** SPP/SCP-set partitioning/covering problem, QAP-quasi-assignment problem and MCF-multi-commodity flow problem. **Solution method:** CG-column generation (based on linear programming relaxation or Lagrangian relaxation) and MH-metaheuristics. **Dataset:** |K|-number of depots, |T|-number of timetabled trips and Test-random or real-world instances. Other abbreviations: B&P-branch-and-price, LP-linear programming and GRASP-greedy randomized adaptive search procedure.

An integrated approach and the traditional sequential approach where the VSP is solved first and then the CSP. The primary objective was to minimize the sum of vehicles and drivers used in the schedule. The proposed integrated approach provided savings of up to one driver when compared to the sequential approach. Similarly, Huisman et al. (2005) showed that the integrated approach has a significant impact when compared to the traditional sequential approach; for an instance with 220 trips, the integrated approach provided a solution with 10 fewer drivers than that of the sequential approach. Borndörfer et al. (2008) used an objective function that is a mix of fixed and variable vehicle cost, fixed cost and paid time of duties and various penalties related to operational requirements of the CSP. For the largest instance, the integrated approach provided an improvement of 3.69% in the objective value when compared to that of the sequential approach. In summary, it has been shown in the literature that simultaneously handling the vehicle and crew aspects is more effective than the traditional sequential approach. An extension of the VCSP is the application of time windows for the timetabled trips, where the departure and arrival times of trips can be shifted within a specified interval (Kéri and Haase, 2008 and Kliewer et al., 2012). Such an extension can be seen as a partial integration of timetabling into the VCSP that offers further flexibility for scheduling vehicles and crews. Kliewer et al. (2012) state that trip shifting enables additional break possibilities between trips for the drivers. Even with very short time windows (up to four minutes) for the timetabled trips, the authors show that enormous savings in the number of planned vehicles and drivers can be achieved.

Integration of the scheduling problem has been explored in other industries as well. In particular, research studies have been carried out in the airline industry to integrate the aircraft routing and crew scheduling problem. Cordeau
et al. (2001) introduce a mathematical model for the complete integration of both the scheduling problems. The model is similar to model (4.17)-(4.22). The authors propose a solution approach that combines column generation and Benders decomposition. The methodology iterates between a Benders master problem that solves the aircraft routing problem and a Benders subproblem that solves the crew scheduling problem, and both the problems are solved by column generation. A heuristic B&B method is used to compute integer solutions. In line with Cordeau et al. (2001), Mercier et al. (2005) propose to model the crew scheduling problem as the Benders master problem and the aircraft routing problem as the Benders subproblem.

4.4 Rescheduling and Robustness

The vehicle and crew schedules are usually computed several months before the actual day-of-operation. However, unforeseen events during operations such as vehicle breakdowns, weather conditions and traffic jams can severely disrupt the planned schedules. Furthermore, in some cases, planned events such as maintenance activities of the infrastructure for a certain period enforce changes to the existing timetable. Therefore, the vehicle and crew schedules may have to be modified as well according to the altered timetable. One area of research in the field of OR is the development of real-time rescheduling methods to reduce the impact of disruptions such as delays or vehicle failures. Visentini et al. (2014) state that much research has been done on the VSP, but considerations on vehicle rescheduling are still relatively unexplored. Furthermore, with its ability to guard against delays, robust planning is receiving more and more attention in the academic literature (Lusby et al., 2018). A solution is said to be operationally robust when the effects of potential delays are minimal (Weide et al., 2010). In this section, methods proposed to recover vehicle and crew schedules are briefly discussed. For a detailed literature review of real-time vehicle schedule recovery methods in bus, railway and airline industries, see Visentini et al. (2014). For a detailed survey on robustness in railway planning problems, see Lusby et al. (2018).

Huisman et al. (2004) present the dynamic vehicle scheduling problem, where the problem considers an environment with significant traffic jams. The authors dynamically reschedule when one or more vehicles that have incurred delays. The authors propose a “cluster-reschedule” heuristic, where the trips are initially assigned to depots using the static VSP and then rescheduled for each depot. Li et al. (2008) consider disruptions due to vehicle breakdowns and one or more vehicles need to be rescheduled to serve the passengers with minimum operating and delay costs. The authors develop a parallel auction algorithm to solve the
vehicle rescheduling problem (VRSP) for a single-depot case. Furthermore, Li et al. (2009) also develop a Lagrangian heuristic approach for the VRSP. The authors extend their previous work to consider trip cancellation costs and more real-world applications. The objective is to keep the changes in the schedule low, because, for example, large changes may make crew rescheduling difficult. Lieshout et al. (2018) extend the VRSP with retiming that allows the possibility to delay trips in order to increase the flexibility of scheduling and avoid trip cancellations. The authors develop a Lagrangian heuristic to solve the VRSP with retiming.

Huisman (2007) define the crew rescheduling problem (CRSP) when the timetables and the vehicle schedules are modified for a certain period due to maintenance activities on large parts of the Dutch railway network. In the CRSP, the initial crew schedule is already given but has become infeasible due to the changes in the underlying timetable and/or vehicle schedule. The objective of the CRSP is to find a feasible schedule with as few modifications as possible. The author proposes a column generation algorithm that is based on Lagrangian relaxation. Potthoff et al. (2010) extend the work carried out by Huisman (2007) to consider the CRSP in a dynamic environment, where the crew needs to be rescheduled as soon as a disruption occurs. Huisman and Wagelmans (2006) integrate the dynamic VSP with crew scheduling and the authors propose two methods. The first one is a sequential approach where the vehicles are scheduled first and then the crews. The other is an integrated approach, which is an adaptation of the column generation approach in Huisman et al. (2005). The integrated approach outperforms the sequential approach. However, the authors discuss various limitations of the dynamic integrated approach when compared to a static version of the VCSP with buffer times between successive trips to safeguard against delay propagation. One of the limitations is the high computation times required by the dynamic approach, and the authors conclude that there is a long way to go before such an approach can be used for practical real-time operations. In the airline industry, Stojković and Soumis (2001) propose a column generation method to recover the aircraft and crew schedules for one day of operations. The authors use a specialized B&B technique to exploit particular characteristics of the problem for obtaining optimal integer solutions. Weide et al. (2010) propose an approach that partially integrates the aircraft routing problem and the CSP. Along with minimizing cost, the authors focus on attaining robust solutions. The two scheduling problems are solved iteratively and the trade-off between cost and robustness is evaluated. The method starts with a minimal cost crew schedule solution without taking aircraft routes into account. At each iteration, the aircraft routing problem is solved taking the current crew schedule solution into account. The CSP is then resolved with the given aircraft routing solution. The iterative process is terminated when the level of robustness cannot be improved any further. By heuristically coupling the two problems, the authors can quickly generate a series of solutions that
incur less cost and are more robust than the solution used in practice. Rescheduling aspects or considerations of robustness for scheduling of electric vehicles have not been reported in the OR literature to the best of our knowledge. Since there are many technological limitations concerning the scheduling of electric vehicles, the development of recovery methods that support practical application of electric vehicles can be seen as a future area of research.

### 4.5 Other Future Research Areas

In this chapter, a detailed literature review on the models and solution methods for the VSP and CSP in public transport was given. The two scheduling problems have been extensively studied in the OR literature. Integration of two or more public transport planning problems is a growing area of research and a literature review on the VCSP was given. A brief discussion on the recovery methods for the vehicle and crew scheduling problems in order to reduce impact of disruptions was carried out. Furthermore, this chapter gave a brief overview of the electric bus technology and its related problems at different planning stages. Electrification of bus fleets in most cities is expected to rise. Due to the limitations and challenges of the electric bus technology, further adjustments have to be made to the current transport planning problems. Therefore, the scheduling of electric vehicles is a crucial and fast growing area of research.

Huisman and Wagelmans (2006) and Weide et al. (2010) are two examples that propose real-time control strategies and robust solution approaches for the integrated vehicle and crew scheduling problem. Such approaches are found to be scarce in the OR literature and the applicability of such approaches in a real-life setting have been hardly reported. Therefore, incorporation of robustness and real-time control strategies in integrated transport planning problems is seen as a future area of research.

The VCSP can be integrated with other transport planning problems. One example is the integration with timetabling that was partially tackled by Kliwer et al. (2012). The succeeding problem of the VCSP is the crew rostering problem (CRP) that has been extensively studied in the OR literature (see e.g. Caprara et al., 1998). Another research area is the integration of the CSP and the CRP (see e.g. Borndörfer et al., 2017). To the best of our knowledge, Mesquita et al. (2013) is the only paper reported in the literature that integrates the VSP, CSP and CRP. Integration of the VCSP with other transport planning problems adds further computational complexity; however such an approach can further improve efficiency of transport systems.
In this chapter, the main developments and findings of this thesis are summarized. Firstly, Section 5.1 gives a brief overview of the three scientific papers in Part II. Section 5.2 discusses the practical impact of each individual paper. Furthermore, the section summarizes the contributions of the papers. Finally, further extensions to the papers are addressed in Section 5.3.

5.1 Overview of Papers

In Paper A, the CSP that involves the use of staff cars is introduced. As mentioned earlier in Chapter 1, a staff car is a company owned car that bus drivers could utilize to travel within the city network during their respective duties. This problem is predominately seen in the public bus transport industry, where the staff cars are often necessary to find a feasible driver schedule. Furthermore, the number of staff cars available at a transport company’s depot is limited. The problem of scheduling the drivers and the staff cars for the drivers is known as the driver scheduling problem with staff cars (DSPSC). This problem is an extension to the CSP and has not been reported in the literature. A MIP model of the DSPSC is given and the paper shows that an integrated solution approach, which simultaneously schedules the drivers and the staff cars, is necessary for
finding feasible solutions. A matheuristic is proposed to solve the DSPSC, where the MIP model is embedded in an ALNS framework.

A column generation approach for solving the DSPSC is proposed in Paper B. The pricing problem that is responsible for generating the driver duties is formulated as a SPPRC, which is solved by a dynamic programming approach that is briefly described in Chapter 3. This paper gives a detailed description of the various labor regulations that transport companies have to consider in the driver scheduling process. Furthermore, all the labor regulations are considered in the SPPRC and the paper analyzes the computational complexity of the domination rules in the label-setting algorithm. Additionally, several heuristic branching strategies to find integer solutions are explored.

Paper C presents the integrated electric vehicle and crew scheduling problem (E-VCSP) that considers the limited driving range and recharging times of electric buses. The problem focuses on handling the electric vehicle and crew scheduling aspects simultaneously in order to minimize the total operational cost. The paper describes two methods for computing lower bounds to the E-VCSP. The traditional sequential approach is also discussed, where the E-VSP is solved first before the CSP, and the approach is used to find upper bounds in short computation times. An ALNS that utilizes B&P heuristic methods is proposed to tackle the E-VCSP. The paper compares the performance of the proposed heuristic to that of the sequential approach.

5.2 Impact in Practice

During the initial stages of the project, it was recognised that several transport companies use staff cars to schedule their drivers. In Paper A, three instances from transport companies in Denmark and Sweden are used to introduce and examine the DSPSC. Moreover, the instances are used to test the developed ALNS heuristic. The column generation method for solving the DSPSC in Paper B is tested on instances from several Northern European transport companies that operate in six cities. More importantly, the developed algorithm captures all the challenges related to the labor regulations that transport companies face during the scheduling process. Papers A and B demonstrate that the DSPSC cannot explicitly be solved by commercial state-of-the-art solvers such as CPLEX and advanced optimization techniques are required to tackle the complexities of large real-life cases. During the project, essential parts of the developed algorithms have been integrated into existing planning tool built for the end users. Additionally, the developed algorithms have been utilized for several pilot projects with transport companies. For example, the developed algorithm provided an
5.2 Impact in Practice

overall improvement of 1.2% when compared to the driver schedule made by manual planners at a transport company that operates in one of the largest cities in Denmark. The improvement represents annual savings of 301,117 €. The aim of this project is also to create further significant positive impact on the transport industry by extending the work carried out in the papers to test more instances from various cities.

Currently, the ALNS heuristic proposed in Paper C to tackle the E-VCSP has not been applied in practice. However, the heuristic is tested on three real-life instances from Denmark and Sweden that contain up to 1,109 timetabled trips. The results indicate that the heuristic can improve the operational efficiency of transport systems. The percentage decrease in the total operational cost provided by the heuristic when compared to the traditional sequential approach is in the range 1.17-4.37% on average. Furthermore, a sensitivity analysis of the electric bus technology is presented and it indicates that the heuristic could also be seen as a strategic tool that supports the transport companies in making crucial decisions such as investment in battery capacities of electric buses and charging infrastructure. Since most cities in Europe have planned to electrify their bus fleet by 2025, the practical relevance of the methods in Paper C is expected to increase significantly in the coming years. Nevertheless, given the benefits of the integrated approach over a sequential approach that practitioners often apply, the heuristic is very likely to be deployed in a decision support system.

Table 5.1 summarizes the contributions of the papers with respect to the research questions formulated in Chapter 1.

<table>
<thead>
<tr>
<th>Research questions</th>
<th>Contributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>How could large-scale vehicle and crew scheduling problems be integrated using optimization algorithms?</td>
<td>An ALNS that utilizes B&amp;P heuristic methods is proposed. It is tested on real-life instances with up to 1,109 timetabled trips (Paper C).</td>
</tr>
<tr>
<td>What is the overall benefit, in terms of operational cost, of integrating vehicle and crew scheduling problems?</td>
<td>By comparing to the traditional sequential approach, the ALNS heuristic in Paper C provides improvements in the range of 1.17-4.37% on average.</td>
</tr>
</tbody>
</table>

Table 5.1: Summary of the contributions of the papers with respect to the formulated research questions.
5.3 Future Work

The individual papers also suggest future research areas in detail. The following points highlight the future work:

- **Practical extensions to the staff car problem**
  In Papers A and B, the staff cars are only allowed to make round trips from the depot to the bus stops. A practical extension to the staff car problem is to incorporate more advanced schedules, where the bus drivers could also use the staff cars to travel between bus stops. A study could be carried out to extend the current work to tackle such a problem and analyze its benefits and computational complexity. Furthermore, the current study did not consider any resources related to the usage of staff cars such as cost and distances. An interesting variation of the staff car problem is the consideration of electric cars and its recharging requirements, which may be highly relevant in the future.

- **Modeling battery charging and discharging behaviour in electric buses**
  In Paper C, only the limited driving range in km of electric vehicles and minimum recharging duration are considered. However, more realistic battery behaviour could be studied and incorporated in the E-VCSP to further analyze the practical implications of electric vehicles. Furthermore, the E-VCSP in Paper C considers only the depot charging system, where drivers are not required to attend to the vehicle when it is being recharged. The E-VCSP could be extended for other charging systems, which may require driver assistance during the recharging activities of vehicles.

- **Tackling large master problems in column generation methods for solving integrated scheduling problems**
  Both Papers B and C demonstrate the increased computational complexity of solving the master problems in column generation methods for the DSPSC and the E-VCSP, respectively. Several acceleration techniques are successfully carried out to speedup the solution process of the pricing problems. In most cases, the computational effort required for solving the master problem is tremendous. For example, in the E-VCSP, an integrated column generation method is used to compute lower bounds and it is found that 66.75% of the total time computation was spent on solving the master problem. For some instances, the optimal LP solutions could not be found within the time limit of 48 hours. Therefore, acceleration techniques and suitable solutions methods have to be explored to solve integrated scheduling problems in reasonable computation times. One solution approach that could be explored is the combination of column
generation and Benders decomposition, which is proposed by Cordeau et al., 2001 and Mercier et al., 2005 to solve the integrated aircraft routing and crew scheduling problem.

In conclusion, this thesis tackles several challenges faced by transport companies. However, with ever changing policies and the introduction of new technologies in the public transport industry, it is believed that new challenges will arise that have to be considered as part of a continuous research process.


Part II

Scientific Papers
A Matheuristic for the Driver Scheduling Problem with Staff Cars

Shyam S.G. Perumal\textsuperscript{a,b}, Jesper Larsen\textsuperscript{a}, Richard M. Lusby\textsuperscript{a}, Morten Riis\textsuperscript{b}, Kasper S. Sørensen\textsuperscript{c}

\textsuperscript{a}Department of Technology, Management and Economics, Technical University of Denmark, Kgs. Lyngby, Denmark  
\textsuperscript{b}QAMPO ApS, Aarhus, Denmark  
\textsuperscript{c}Trapeze Group Europe, Aarhus, Denmark

1 Abstract: In the public bus transport industry, it is estimated that the cost of a driver schedule accounts for approximately 60\% of a transport company’s operational expenses. Hence, it is important for transport companies to minimize the overall cost of driver schedules. A duty is defined as the work of a driver for a day and the driver scheduling problem (DSP) is concerned with finding an optimal set of driver duties to cover a set of timetabled bus trips. Numerous labor regulations and other practical conditions enforce drivers to travel

1Published in European Journal of Operational Research, May 2019
within the city network to designated bus stops to start/end duty, to take a break or to takeover a bus from another driver. This paper focuses on the driver scheduling problem with staff cars (DSPSC), where staff cars can be utilized by the drivers to fulfill their travel activities. However, staff cars should always be returned to the depot and can perform multiple round trips during the day. The problem is restricted by the number of cars available at the depot. We present a matheuristic for solving the DSPSC and the proposed method is tested on instances from Danish and Swedish companies. A comparison with a state-of-the-art mixed integer programming (MIP) solver indicates that the matheuristic provides better solutions, with comparable computation times, for 6 out of 10 large instances. For instances that have more than 6 staff cars and 1200 bus trips, the improvement is 13-15% on average.

**Keywords:** Transportation, Driver scheduling problem, Heuristics, Adaptive large neighborhood search

### A.1 Introduction

Growing populations in cities worldwide demand well-organized public transport systems that prevent long travel times, traffic congestion, road accidents and pollution. Public transport systems are considered to be the backbone of sustainable urban development. The passengers expect a high level of service; i.e., the transport system should be accessible, comfortable, affordable and it should be possible to reach destinations quickly. The objective of transport companies is to provide high quality service to the passengers while minimizing their overall operational cost (Desaulniers and Hickman, 2007 and Ibarra-Rojas et al., 2015). Transport companies are constantly faced with the challenge of planning for cities with large scale transport systems. Government and EU policies, labor regulations and other practical conditions further challenge transport companies to efficiently utilize their resources. As a consequence, over the years, there has been an increase in the development of decision support tools based on mathematical programming approaches to aid transport companies in planning (Desrochers and Soumis, 1989, Wren et al., 2003, Smith and Wren, 1988 and Lourenço et al., 2001). Typically, the transport planning process involves solving several planning subproblems as it is too complex to solve the entire planning problem in one integrated step. The planning subproblems include Timetabling, the Vehicle Scheduling Problem (VSP), the Driver Scheduling Problem (DSP) and the Driver Rostering Problem (DRP). Public authorities often define the timetabled trips, where the arrival and departure times at all bus stops in a
city network are determined. The timetabled trips are utilized by transport companies to schedule their buses and drivers. The VSP assigns buses to the timetabled trips such that every trip is covered by a bus and the objective is to minimize the operational cost based on bus usage. A bus typically covers a sequence of trips from the time it leaves the depot until it returns to the depot. A driver duty is defined as the work of a driver for a day and the DSP is concerned with finding an optimal set of duties that covers all bus trips. Given a set of generic duties over a certain time horizon, e.g. a month, the DRP assigns these duties to the available drivers.

In the DSP, transport companies have to plan driver schedules based on known bus schedules. In most cases, the bus schedules are different on weekdays, weekends and holidays. Hence, transport companies need to create driver schedule for each of these bus schedules. While determining the driver schedule, companies must consider two important aspects: minimizing the cost and ensuring feasibility of driver duties with respect to various labor regulations. Most commonly, the cost comprises of wages paid to the drivers, which is known to be more than the operational expenses of buses. It is estimated that the cost of a driver schedule accounts for approximately 60% of a transport company’s operational expenses. Hence, a small improvement in the cost of driver schedule can lead to large savings. For a Danish transport company with over 600 drivers, it was shown that a reduction in cost of 1.2% represents 2-2.5 million DKK in savings in a year. Furthermore, the transport companies have to strictly abide by the labor rules and regulations. The three most common rules that apply for all transport companies are maximum working time for a driver, minimum/maximum number of breaks, and maximum time between breaks. These rules forbid the driver from driving a bus over a prolonged period and allow the driver to have sufficient breaks during the day. Hence, a driver may cover only a few consecutive bus trips before the driver takes a break or is relieved of duty. Another important condition for transport companies is the computation time required to build a driver schedule. Planning departments of transport companies with their experience in the bus transport industry and knowledge of the city network take three to four weeks to manually create a new driver schedule. A decision support tool based on mathematical programming approaches potentially eliminates the resources and the time required to plan. Therefore, a decision support tool that provides optimal or near-optimal solutions in quick computation time would assist transport companies to create and operate efficient transport systems.

Transport companies must also consider other practical limitations in the city network during the planning process. These limitations are concerned with the characteristics of the bus stops. Only at certain bus stops can the driver be allowed to sign on/off duty or to handover the bus to another driver. Furthermore, the driver is allowed to have a break only at certain bus stops, which is
dependent on the availability of facilities such as restroom and canteen. Figure A.1 illustrates an example of a duty where the driver covers a few trips. As it can be seen in the figure, the driver usually has to travel during the day of work within the city network to sign on/off duty, visit a bus stop where taking a break is allowed or takeover a bus from another driver. If the distance between the bus stops is short, then the driver can travel by foot. The driver can also travel as a passenger on a bus to designated bus stops. However, in some cases, a bus stop can only be reached feasibly with use of a car. Transport companies usually have a fleet of cars, which are defined as staff cars that the drivers could utilize. A driver, as part of his/her travel activity, usually takes a staff car from the depot to visit another bus stop and parks the staff car at the visited bus stop. Another driver may utilize the parked car from the same bus stop and drive it back to the depot as part of his/her travel activity during the duty. A round staff car trip is defined as the combination of a departure trip from the depot and an arrival trip to the depot. A staff car can perform multiple round trips during the day; however, a staff car should always be returned to the depot and only the limited cars that are available at the transport company’s fleet can be used to fulfill the travel activities of the drivers. Simultaneously, scheduling the drivers and the staff cars for the drivers gives rise to the driver scheduling problem with staff cars (DSPSC), which is an extension of the DSP. Such problems have not been reported in the Operations Research (OR) literature to the best of our knowledge. The DSP is a very complex problem, similar structured problems have been proven to be $\mathcal{NP}$-hard problems (Fischetti et al., 1987), and the DSPSC adds further complexity to the DSP.

Heuristic algorithms that are designed by combining metaheuristics and mathematical programming techniques are known as matheuristics (Boschetti et al., 2009). In this paper, we propose a matheuristic for solving the DSPSC. A mixed integer programming (MIP) model, exact method, is embedded in an adaptive large neighborhood search (ALNS) heuristic. Large neighborhood search (LNS) was proposed by Shaw (1998) and the author applied the heuristic to vehicle routing problems. LNS is based on a local search framework such as simulated annealing (SA) or hill climber, where an initial solution is gradually improved by iteratively destroying and repairing the solution using a destroy and a repair method respectively. ALNS was proposed by Ropke and Pisinger (2006) where multiple destroy and repair methods are defined within the same search. Each destroy and repair method is assigned a weight that controls the selection of the particular method during the search. Problems, such as the DSP, where Dantzig-Wolfe Decomposition has been used with success are good candidates for LNS and ALNS heuristics (Pisinger and Ropke, 2019). Exact approaches are known to be highly effective for small to medium sized instances of hard problems but are inefficient for large instances; hence, heuristics are commonly used in practice (Jourdan et al., 2009 and Blum et al., 2011). However, metaheuristics based on a local search framework are often ineffective for highly con-
strained problems where feasible areas of the solution space are disconnected (Dumitrescu and Stützle, 2003). Muller et al. (2012) designed a matheuristic based on ALNS for solving the lot sizing problem (LSP) with setup times. The authors’ motivation for using a MIP model for the repair phase of the ALNS heuristic was that it could tackle the challenges of constructing and exploring a neighborhood of a given solution. MIP models can be very effective for exploring large neighborhoods within a local search procedure and guide the search to move between feasible regions of the solution space (Jourdan et al., 2009). Since the DSPSC is considered to be a tightly constrained problem, where even finding a feasible solution could be challenging, the matheuristic approach is viewed as a powerful optimization method for solving it. In 2009, Jourdan et al. (2009) stated that there has been an increase in number of works carried out on matheuristic approaches. The approach’s ability to simultaneously exploit advantages of heuristics and exact methods has led to obtaining best solutions for most practical problems.

Trapeze Group Europe A/S (TGE) is an international provider of decision support tools within planning and operations for both public and private transport companies. Real-life instances of the DSPSC from a Danish and a Swedish transport company were acquired from TGE’s system for this paper. Since DSPSC is a very realistic problem, this paper primarily contributes to research areas...
within traditional DSP and within applications of OR techniques for improving efficiency of transport systems.

This paper is organized as follows. Section A.2 gives a description of the existing literature related to the DSP. In Section A.3 we provide a formal description of the DSPSC with the help of a mathematical model. Section A.4 introduces the proposed matheuristic framework for solving the DSPSC. The section also gives an outline of a greedy heuristic that provides an initial solution. Section A.5 details the computational study based on experimental tests performed on instances from Danish and Swedish transport companies. Finally, Section A.6 concludes the paper and addresses future directions of research. We also briefly discuss the challenges of integrating mathematical programming approaches such as the proposed matheuristic into decision support tools.

### A.2 Related Literature

Common formulations of the DSP are based on set partitioning/covering problem (SCP), where the formulation is used as a duty selection module with the selected duties covering all bus trips at minimum cost (Ibarra-Rojas et al., 2015). To find the optimal solution, all the feasible duties have to be considered in the SCP formulation. Due to potentially being a large number of possible duties, the formulation is intractable by exhaustive enumeration techniques. Some authors, e.g. Smith and Wren (1988) and Wren et al. (2003), have considered reducing the number of duties being generated before solving the SCP. Smith and Wren (1988) heuristically generate a feasible subset of duties for the SCP. The SCP is solved by relaxing the integrality constraints and an integer solution is found using a branch-and-bound algorithm. The algorithm terminates when the current integer solution is within 0.5% of the integer optimum. The authors had developed the method as part of a commercial software system and reported some of the experiments run on the system. The largest instance included 309 constraints (bus trips), 4892 variables (duties) in the SCP formulation and the best integer solution was found in 238 seconds. Similar to Smith and Wren (1988), Wren et al. (2003) solve the SCP by considering only a subset of feasible duties. A large set of potential driver duties is generated and refined by heuristic procedures. The authors focused more on combining theory and practice to create a user-friendly and flexible system. Portugal et al. (2009) presented SCP based models that were developed in collaboration with planners and end-users of several transport companies in Portugal. The authors aimed at developing models to produce solutions that could be applied in real situations and, hence solution quality was not the only criteria for evaluating models. The authors tested instances with up to 347 bus trips and 23305 duties in the SCP
formulation.

Exact approaches such as Branch & Price, where SCP implicitly considers all the possible duties, are commonly used in the literature for generating duties. Desrochers and Soumis (1989) devised a column generation method for solving real-life instance of a transport company operating in an American city that had a fleet of 25 buses. The resulting SCP formulation had 167 bus trips. However, solving large scale instances of the DSP by column generation approaches is notorious for being computationally expensive due to the need to solve resource constrained shortest path problem (RCSPP) at every iteration (Wren et al., 2003 and Ibarra-Rojas et al., 2015). Yunes et al. (2005) devised a hybrid column generation approach that used constraint programming (CP) to generated feasible duties. The authors also reported that solving the RCSPP by dynamic programming techniques suggested by Desrochers and Soumis (1989) was computationally inefficient for large instances. The authors tested the two aforementioned methods on instances from a bus company in the city of Belo Horizonte, Brazil. The sizes of the instances varied from 10 to 210 bus trips. The column generation algorithm based on dynamic programming could not solve instances more than 90 trips within a time limit of 24 hours. It was reported that 90% of the total computation time, on average, was spent on solving the RCSPP. However, the column generation based on CP was able to solve the largest instance of 210 trips in less than 15 hours. Mauri and Lorena (2007) applied a metaheuristic method known as the population training algorithm (PTA), a derivation of the genetic algorithm (GA), as part of the column generation framework to generate feasible duties. The authors tested the method on randomly generated instances that were based on real problems of a Brazilian transport company and the sizes of the instances varied from 25 to 500 bus trips. The authors compared the proposed method to a SA approach and the results for the largest instance indicated that the proposed method was faster than the SA approach by a factor 20 with an improvement of 0.15% in solution quality. Li et al. (2015) proposed a column generation approach that is based on a hyper-heuristic, which is similar to the idea of having multiple repair heuristics in a ALNS. The authors generate all feasible duties for a given instance and several heuristics (local search, swap heuristic and greedy based heuristic) are devised to select a subset of feasible duties at each iteration of the column generation framework. In the method proposed by Li et al. (2015), the hyper-heuristic evaluates the different heuristics based on their selection of duties that contribute to the improvement of the objective at each iteration of the column generation framework. The authors tested the method with instances provided by Mauri and Lorena (2007) and the largest instance with 500 bus trips had a total of 8.4 million feasible duties. The column generation based on hyper-heuristics yielded solutions that, on average, had a gap of 2.12% from the best known solutions, which were provided by the method proposed by Mauri and Lorena (2007). However, for the largest instance, the hyper heuristic was
Heuristic approaches, if designed well, are known to provide good solutions in reasonable computation time for large scale problems. Lourenço et al. (2001) proposed multiobjective metaheuristics for solving real-life instances of the DSP. In most formulations of the DSP, the objective is to minimize cost of driver duties. However, in practice, different transport companies take several objectives into account while planning. Hence, the authors solved the DSP involving multiple objectives such as minimizing total number of duties, minimizing number of bus changes and minimizing number of over-covered bus trips. A large number of duties that comply with the labor regulations and the transport companies' rules is heuristically generated for the SCP formulation. The SCP is solved by metaheuristic methods tabu search (TS) and GA. The authors also devised a greedy randomized adaptive search procedure (GRASP) for solving a subroutine of the TS and GA. The proposed methods were tested on instances with up to 348 bus trips and 74000 duties in the SCP formulation. The devised GA provided solutions that were comparable to that of linear programming (LP) based algorithm. For the largest instance, the computation time of the GA was approximately 5 times shorter than that of the LP based method. Similarly, Li and Kwan (2003) solved the DSP by GA. The authors tested the algorithm on instances with up to 1873 trips and 50000 duties in the SCP formulation. The proposed method was able to solve the largest instance in 1350 seconds, which had a gap of 3.89% from the best known solution provided by a MIP solver with a computation time of more than 10 hours. De Leone et al. (2011a) devised a GRASP for solving the DSP for instances with up to 161 bus trips from a Italian transport company. The proposed method was compared with an exact method that uses a MIP solver to solve the SCP model with up to 2 million feasible duties. The results showed that the exact method took more than 3 hours to find a feasible integer solution for the largest instance, whereas the GRASP was able to provide a solution in one minute that was almost 30% better than the first feasible solution provided by the exact method. The authors, De Leone et al. (2011b), also compared the devised GRASP heuristic to a hybrid of GRASP and variable neighborhood search (VNS). The results indicated that the hybrid version provided improved solutions with comparable computation times. For the largest instance, improvement was found to be 5.2%. Ma et al. (2016) proposed a VNS heuristic for solving the DSP with instances from a transport company in Beijing, China that had up to 501 bus trips. The authors compared solutions from the proposed method to that of the solutions manually created by the planners in the company. For the largest instance, the VNS heuristic provided, on average, an improvement of 6% with a computation time of 22 minutes, whereas almost five hours was taken to create the manual plan.

Most of the works published in the literature have aimed at developing methods to solve large sized instances of the DSP, which include complexities that arise
in practice. Since the DSP is a highly relevant problem in the public transport industry, some have considered implementing the methods as part of a commercial software system. However, none of the published works address the DSPSC. Several researchers in the 1980s (Ball et al., 1983 and Darby-Dowman, 1988) recognized the need to integrate the VSP and the DSP (IVDSP), where scheduling of buses and drivers are simultaneously carried out. The integration of the two scheduling problems could lead to further cost reductions and efficiency gains for transport systems (Freling et al., 2003). However, the IVDSP has received very little attention in the literature due to its increased complexity of formulating and handling large real-life instances that require immense computation times to be solved (Borndörfer et al., 2008 and Huisman et al., 2005). Combining lagrangian relaxation and column generation have commonly been used to integrate the problems, e.g. Huisman et al. (2005), and the IVDSP is known to be a growing area of research. By introducing the DSPSC, we believe that there will be implications on the IVDSP, which has to be studied further.

A.3 Problem Description and Mathematical Modelling

This section presents the mathematical model which serves as the formal description of the problem. Let $T$ be the set of bus trips that need to be covered and let $D$ be the set of all valid duties that comply with the labor regulations and the company’s operational rules. Each $d \in D$ is checked to see if it requires one or more staff cars as part of its travel activities and all car travels are grouped into set $C$. Let $N$ denote the set of nodes that drivers could visit using a staff car and $r \in N$ is denoted as the car depot. Each car travel, $i \in C$ has a departure node $k_i$, an arrival node $l_i$, departure time $u_i$ and arrival time $v_i$. A departure car travel is defined as a car travel that departs from depot $r$. Set of departure car travels is denoted as $\hat{C} \subset C$, where $k_i = r$, $l_i = n \in N \setminus \{r\}$ and $i \in \hat{C}$. Similarly, an arrival car travel is defined as a car travel that arrives at depot $r$. Set of arrival car travels is denoted as $\check{C} \subset C$, where $k_i = n \in N \setminus \{r\}$, $l_i = r$ and $i \in \check{C}$. The cost or paid time associated with duty $d \in D$ is represented as $c_d$. Binary matrix $A$ is defined, where $a_{td}$ is 1 if duty $d \in D$ is covering bus trip $t \in T$ and 0 otherwise. Another binary matrix $G$ is defined, where $g_{id}$ is 1 if duty $d \in D$ utilizes car travel $i \in C$ and 0 otherwise. We define a car match as a combination of a departure car travel from the depot and an arrival car travel to the depot to form one round trip as depicted in Figure A.2. Binary matrix $H$ is defined, where $h_{ij}$ indicates whether two car travels can be matched as one round trip, i.e. $l_i = k_j$, $v_i \leq u_j, i \in \hat{C}$ and $j \in \check{C}$. The time a staff car is idle at a node other than the depot is defined as car idle time. To
simplify the car matches notation, we define $H$ as the set of all car matches, i.e. $H = \{(i, j) \mid h_{ij} = 1\}$.

![Staff Car Match Diagram](image-url)

**Figure A.2:** Staff Car Match. Car travel $i$ of duty $d$ departs from depot $r$ to node $n$ and car travel $j$ of duty $w$ arrives at depot $r$ from node $n$. The idle time of the staff car at node $n$ is calculated as $(u_j - v_i)$. The figure illustrates an example of a round trip.

Four decision variables are defined in the mathematical model. Binary variable $x_d$ indicates if duty $d \in D$ is selected or not and binary variable $y_t$ indicates if a bus trip $t \in T$ remains uncovered or not. A penalty of $\beta$ is incurred if a bus trip is uncovered. Binary variable $z_i$ indicates if car travel $i \in C$ is used or not and binary variable $s_{ij}$ indicates if car travel $i \in C$ is matched with car travel $j \in C$ to form one round trip. The maximum number of staff cars that is available at the depot is denoted as $Q$.

As depicted in Figure A.3, a staff car can perform multiple round trips during the day. To estimate the number of staff cars that are being used at a particular time, we define $O$ as the set of all departure times of a staff car from the depot, i.e. $\{u_i\}$ where $i \in C$, and $P_o$ as the set of all possible car matches that are active at time $o$,

$$P_o = \{(i, j) \mid (i, j) \in H \land u_i \leq o \land v_j \geq o\} \quad \forall o \in O \quad (A.1)$$

The mathematical formulation of the DSPSC is as follows,

$$\text{Minimize} \sum_{d \in D} c_d \cdot x_d + \beta \sum_{t \in T} y_t \quad (A.2)$$
subject to:

\[
\sum_{d \in D} a_{td} \cdot x_d + y_t \geq 1 \quad \forall t \in T \quad (A.3)
\]
\[
\sum_{d \in D} g_{id} \cdot x_d \leq z_i \cdot M \quad \forall i \in C \quad (A.4)
\]
\[
\sum_{d \in D} g_{id} \cdot x_d \geq z_i \quad \forall i \in C \quad (A.5)
\]
\[
\sum_{j \in \hat{C}} h_{ij} \cdot s_{ij} = z_i \quad \forall i \in \hat{C} \quad (A.6)
\]
\[
\sum_{i \in \hat{C}} h_{ij} \cdot s_{ij} = z_i \quad \forall j \in \hat{C} \quad (A.7)
\]
\[
\sum_{(i,j) \in P_o} s_{ij} \leq Q \quad \forall o \in O \quad (A.8)
\]
\[
x_d \in \{0, 1\} \quad \forall d \in D \quad (A.9)
\]
\[
y_t \in \{0, 1\} \quad \forall t \in T \quad (A.10)
\]
\[
z_i \in \{0, 1\} \quad \forall i \in C \quad (A.11)
\]
\[
s_{ij} \in \{0, 1\} \quad \forall i \in \hat{C}, \forall j \in \hat{C} \quad (A.12)
\]

The objective (A.2) is to minimize the overall cost of driver duties and the penalty for leaving a bus trip uncovered. Constraints (A.3) ensure that a bus
trip is covered by at least one duty or is left uncovered. Constraints (A.4) ensure that a car travel is selected if it is utilized by one or more duties in the final schedule. $M$ is a large number and can be set as the seating capacity of the staff cars. Constraints (A.5) ensure that a car travel is not selected if none of the duties in the final schedule utilize it. Constraints (A.6) together with constraints (A.7) ensure that a selected departure car travel from the depot is matched with an arrival car travel to the depot to form one round car trip. Constraints (A.8) ensure that at all times during the day the number of staff cars being utilized is not more than the maximum number of staff cars available at the depot.

A.4 Solution Method

A.4.1 Greedy heuristic

To construct an initial solution, a basic greedy heuristic is implemented. A duty is selected based on an evaluation function and added to the solution at each iteration of the greedy heuristic. For the DSPSC, two evaluation functions are applied; one for selecting duties and another for selecting car matches. The duties are evaluated based on a function $\Delta$ and we determine $\Delta_d = c_d + \beta(I_d - J_d + K_d)$, where $I_d$ is the number of trips covered by duty $d$ that are already covered in the solution, $J_d$ is the number of trips covered by duty $d$ that are not covered in the solution and $K_d$ is the number of cars travels used by duty $d$ (i.e. $\sum_{i \in C} g_{id}$). At each iteration, the values of $I_d$ and $J_d$ are adapted based on the trips being covered in the solution and duty with minimum $\Delta_d$ is the best candidate to be added to the solution. The newly inserted duty potentially involves car travels and they have to be matched. For instance, duty $d$ might have a departure car travel from depot $r$ to node $n$ and hence would need a matching car travel that would return the car from node $n$ to depot $r$ to form one complete round trip. In the greedy heuristic, the matching car travel is selected based on minimum idle time of the staff car at node $n \in N \setminus \{r\}$. By returning the staff car quickly back to the depot, it has the possibility of being used for multiple round trips. Hence, improving the utilization of the staff cars is the underlying motivation for minimizing car idle time.

The greedy heuristic procedure is shown in Algorithm 5. At each iteration of the greedy heuristic, a candidate list, $q$, of duties is created and added to the solution $s$. $E$ in Line 5 denotes the set of unmatched car travels in the $s$ and is updated (Line 20) when car travels are matched or new unmatched car travels are added to $s$. For each $i \in E$, all its matching car travels, i.e all the car travels that would form a round car trip with car travel $i$, are collected in set
F. From set $F$, car travel $j$ that forms a round car trip with minimum car idle time, $\text{carIdleTime}(i,j)$, is selected (Line 10). Set $G$ consists of all the duties with car travel $j$ (Line 11) and duty $w = \arg \min \in G(\Delta_w)$ is selected (Line 12) and added to $q$ (Line 13). In some cases during the advancement of the heuristic, it might select a duty with a car travel that cannot be matched. In such circumstances, the heuristic removes the selected duty from $\mathcal{D}$, empties $q$ and is forced to select the next best duty in terms of $\Delta$ (Line 15 - Line 17). The loop (Line 6 - Line 21) terminates when no more unmatched car travels exist in the partial solution, i.e. set $E$ is empty.

The greedy heuristic terminates when all the bus trips have been covered and all the car travels in $s$ have been matched. Even though the heuristic tries to improve the utilization of a staff car through function $\text{carIdleTime}()$, it cannot control the number of cars being used. Hence, the greedy heuristic often builds an initial solution which does not satisfy the maximum number of cars condition.

\begin{algorithm}
\caption{Greedy heuristic}
\begin{algorithmic}[1]
\State \textbf{Initialization:} $s \leftarrow \emptyset$, $q \leftarrow \emptyset$;
\While {stop criterion not met}
\State $d \leftarrow \arg \min \in \mathcal{D}(\Delta_d)$;
\State $q \leftarrow q \cup \{d\}$;
\State Set $E = \{i \mid g_{id} = 1, i \in \mathcal{C}\}$;
\While {$E \neq \emptyset$}
\For {$i \in E$}
\State Set $F = \{j \mid (i,j) \in \mathcal{H}, j \in \mathcal{C}\}$;
\If {$F \neq \emptyset$}
\State $j \leftarrow \arg \min \in F(\text{carIdleTime}(i,j))$;
\State Set $G = \{w \mid g_{jw} = 1, w \in \mathcal{D}\}$;
\State $w \leftarrow \arg \min \in G(\Delta_w)$;
\State $q \leftarrow q \cup \{w\}$;
\Else
\State $\mathcal{D} \leftarrow \mathcal{D} - \{d\}$;
\State $q \leftarrow \emptyset$;
\State \textbf{go to} 3;
\EndIf
\EndFor
\EndWhile
\State \textbf{Update} $E$;
\State $s \leftarrow s \cup q$;
\State $q \leftarrow \emptyset$;
\EndWhile
\State \textbf{return} $s$
\end{algorithmic}
\end{algorithm}
A.4.2 Matheuristic

In our matheuristic setting, the mathematical model described in Section A.3 is embedded in an ALNS framework to obtain high-quality solutions in reasonable computation time.

A neighborhood is defined as the set of neighboring solutions of a current solution and a local search procedure iteratively moves the current solution to a neighboring solution. In LNS, the neighboring solutions could be reached by applying a destroy method and then a repair method to the current solution. Hence, the neighborhood is implicitly defined by the destroy and repair methods. In ALNS, multiple destroy and repair methods are applied and hence different neighborhoods can be explored within the same search. Each of the destroy and repair methods is assigned a modifiable weight which is updated based on the performances of the methods during the course of the search. Ropke and Pisinger (2006) state that not all destroy and repair methods perform equally well and that, for example, one method might be very well-suited for one type of instance and another method might be well-suited for another instance. The diverse and robust nature of the ALNS heuristic has led to its gain in popularity in recent years and has been applied to a large selection of different optimization problems. Some applications include the capacitated arc-routing problem (Laporte et al., 2010), the vehicle routing problem (Pisinger and Ropke, 2007) and the patient admission scheduling problem (Lusby et al., 2016).

Algorithm 6: Adaptive Large Neighborhood search

1 **Initialization:** $s \leftarrow \text{InitialSolution}()$, $s^* \leftarrow s$;
2 $\rho \leftarrow \text{InitializeMethodWeights}()$;
3 **while** stop criteria not met **do**
4 \hspace{1em} Select destroy and repair methods $\mu \in \tau^-$ and $\gamma \in \tau^+$ using $\rho$;
5 \hspace{1em} $s' \leftarrow \text{Destroy}(s, \mu)$;
6 \hspace{1em} $s' \leftarrow \text{Repair}(s', \gamma)$;
7 \hspace{1em} **if** Accept($s$, $s'$) **then**
8 \hspace{2em} $s \leftarrow s'$;
9 \hspace{1em} **end**
10 \hspace{1em} **if** $f(s') < f(s^*)$ **then**
11 \hspace{2em} $s^* \leftarrow s'$;
12 \hspace{1em} **end**
13 \hspace{1em} $\rho \leftarrow \text{UpdateMethodWeights}()$;
14 **end**
15 **return** $s^*$

Algorithm 6 outlines the ALNS procedure where $s$ denotes the current solution,
s' is the neighboring solution and s* is the best solution. The set of all destroy methods is denoted as $\tau^-$ and the set of repair methods is denoted as $\tau^+$. As shown in Line 4, at each iteration of the heuristic a destroy method $\mu \in \tau^-$ and a repair method $\gamma \in \tau^+$ are selected to perform an operation on the current solution. The selection of the methods are dependent on the weights of the methods, $\rho^\mu$ and $\rho^\gamma$, which are dynamically updated during the execution of the heuristic. Well-performing methods have a high weight and thus would have a higher probability of being selected. The probability of a destroy method being selected is determined as,

$$
\zeta_\mu = \frac{\rho^\mu}{\sum_{l \in \tau^-} \rho^l} \quad \forall \mu \in \tau^-
$$

(A.13)

Similarly, the probability of selecting a repair method is determined as,

$$
\zeta_\gamma = \frac{\rho^\gamma}{\sum_{l \in \tau^+} \rho^l} \quad \forall \gamma \in \tau^+
$$

(A.14)

The selection of the destroy and the repair method is made based on a roulette wheel principle using the probabilities calculated in Equations (A.13) and (A.14). The entire search is divided into $n_{seg}$ segments and each segment is defined by $n_{iter}$ iterations. At the end of each segment, the weights of the methods are updated, as shown in Line 13. For each destroy method $\mu \in \tau^-$ and repair method $\gamma \in \tau^+$, $\Omega^\mu$ and $\Omega^\gamma$ define the accumulated score. The number of times a destroy method and a repair method have been selected during a segment is given by $\nu^\mu$ and $\nu^\gamma$ respectively. At each iteration of the heuristic, a score of $\psi$ is awarded to the chosen destroy and repair method and added to $\Omega^\mu$ and $\Omega^\gamma$. The score is given based on the quality of the solution obtained and could be one of $\psi_1, \psi_2, \psi_3$ or $\psi_4$. The description of the score parameters is shown in Table A.1, where $\psi_1 > \psi_2 > \psi_3 > \psi_4$. The weights of destroy and repair methods are initialized to 1 and after each segment, the weights are updated as follows,

$$
\rho^\mu = (1 - \lambda) \cdot \rho^\mu + \lambda \cdot \frac{\Omega^\mu}{\nu^\mu} \quad \forall \mu \in \tau^- \quad (A.15)
$$

$$
\rho^\gamma = (1 - \lambda) \cdot \rho^\gamma + \lambda \cdot \frac{\Omega^\gamma}{\nu^\gamma} \quad \forall \gamma \in \tau^+ \quad (A.16)
$$
At the start of each segment, $\Omega^\mu$, $\Omega^\gamma$, $\nu^\mu$ and $\nu^\gamma$ are set to 0. $\lambda \in [0, 1]$ is known as the reaction factor which controls the changes in weights. If $\lambda = 1$ then the roulette wheel selection is only based on the scores of the most recent segment and if $\lambda = 0$ then the weights are kept constant at the initial level. The past performances of the methods are taken into account when $0 < \lambda < 1$. The weight of a method remains unchanged if it was not selected in the segment.

<table>
<thead>
<tr>
<th>Score($\psi$)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi_1$</td>
<td>if the new solution is a new best solution</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>if the new solution is better than the current solution</td>
</tr>
<tr>
<td>$\psi_3$</td>
<td>if the new solution is accepted</td>
</tr>
<tr>
<td>$\psi_4$</td>
<td>if the new solution is rejected</td>
</tr>
</tbody>
</table>

Table A.1: Score parameters for ALNS

Shaw (1998) proposed a hill climber accept criterion in a LNS framework where only improving solutions are accepted. This acceptance criterion, however, has the tendency to get trapped in a local optimum. To diversify the search, solutions that are worse than the current solution should be accepted occasionally. Hence, a score of $\psi_3$ is rewarded to methods that are able to visit unexplored solution spaces. One approach of introducing diversification to the search procedure is the simulated annealing (SA) acceptance criterion which has been successfully used in an ALNS framework by some authors (see e.g. Ropke and Pisinger, 2006 and Lusby et al., 2016). Given a current solution $s$, a worse solution $s'$ is accepted with a probability of $\exp \left(-\frac{f(s') - f(s)}{\theta}\right)$, where $\theta > 0$ is the temperature. The heuristic starts with an initial temperature, $\theta = \theta_{start}$, and this is gradually decreased during the course of the heuristic with the aid of a cooling factor $\alpha \in (0, 1)$. The temperature is decreased to $\theta = \theta \cdot \alpha$ at the end of each segment. During the last iterations of the ALNS, worse solutions are unlikely to be accepted and hence the framework behaves like a hill climber. Similar to authors Ropke and Pisinger (2006), we determine $\theta_{start}$ based on the problem instance at hand, i.e. $\theta_{start}$ is set such that a solution $\omega$% worse than the initial solution is accepted with probability 0.5. In our case, the initial solution is obtained from the greedy heuristic for the DSPSC described in Section A.4.1. However, the greedy heuristic does not ensure feasibility with respect to the number of staff cars being used. In order to avoid setting $\theta_{start}$ too large, the penalty incurred for violating the maximum number of staff cars constraint (A.8) is disregarded. The heuristic terminates when it has performed $n_{seg}$ segments.

Because the greedy heuristic often results in an infeasible solution, the ALNS heuristic is restarted when it reaches the first feasible solution. When restarting the heuristic, the weights of the methods are also reinitialized and $\theta_{start}$ is recalculated based on the first feasible solution. However during the execution
of the heuristic, we allow it to visit infeasible regions of the solution space. It is believed that local search heuristics often have difficulties in moving from one promising area of the solution space to another in tightly constrained problems (Ropke and Pisinger, 2006). To tackle this, some authors, e.g. Cordeau et al. (2001) and Lourenço et al. (2001), allow the search to visit infeasible solutions by relaxing some constraints. Similarly, in addition to the maximum number of staff cars constraint (A.8), we relax the car matching constraints (A.6) and (A.7) to allow for unmatched car travels in the solution; however, a penalty is added to the objective function when a car travel is unmatched.

### A.4.2 Destroy Methods

Given a solution, let $\bar{D}$, $\bar{C}$ and $\bar{H}$ denote the set of duties, car travels and car matches in the solution respectively. Sets $D'$, $C'$ and $H'$ denote the duties, car travels and car matches not in the solution. For the DSPSC, we propose three destroy methods and these are as follows,

1. Random removal of duties
   
   To diversify the search, random duties are removed from $\bar{D}$ and the set of removed duties is denoted as $D^R$. The number of duties to be removed, $|D^R|$, is controlled by the degree of destruction parameter, $\xi$, and is determined as $1 \leq |D^R| \leq \xi \cdot |\bar{D}|$. The car travels to be removed from the solution are dependent on the duties removed from the solution, i.e. $C^R = \{i \mid g_{id} = 1, i \in \bar{C}, d \in D^R\}$, and the car matches to be removed are determined as $H^R = \{(i, j) \mid (i, j) \in \bar{H} \land (i \in C^R \lor j \in C^R)\}$. In most cases, when car travels and car matches are removed from the solution, the destroyed solution consists of unmatched car travels. For instance, a car travel $i$ is in the solution but its matching car travel $j$ was removed from the solution. The set of unmatched car travels in the destroyed solution is represented as $C^U = \{i \mid (i, j) \in H^R, i \in \bar{C}, j \in C^R\}$.

2. Worst removal of duties
   
   The function $\Delta$, as described in Section A.4.1, prefers duties that cover many of the bus trips with minimum overcoverage and car travels. Hence, as part of intensification strategy, duty $d$ given by $\arg \max_{d \in D}(\Delta_d)$ is likely to be removed from the solution. Duties in the solution, $\bar{D}$, are sorted in descending order of $\Delta_d$. The duties to be removed are determined as, $D^R = D^R \cup D[g^B \cdot |\bar{D}|]$ where $g$ is a random number in the interval $[0,1)$ and $B \geq 1$ is a degree of randomization parameter that controls the randomness in the selection of the duties. A low value of $B$ corresponds to random selection of duties and a high value corresponds to selecting duty with highest $\Delta_d$ value. The number of duties to be removed, $|D^R|$ is
determined in the similar manner as the random removal of duties using the $\xi$ parameter.

3. Random removal of car travel matches
The aforementioned destroy methods do not specifically target the car schedule in the solution where there are probably a larger number of staff cars being used than strictly required. The methods leave partial car matches in the solution and hence do no guarantee making significant changes in the car schedule. To address this issue, car matches in the solution, $\hat{H}$, are randomly selected and removed from the solution with the objective of reducing the number of cars. The set of car matches to be removed is denoted as $\mathcal{H}^R = \{(i, j) \mid (i, j) \in \hat{H}\}$ and is also controlled by the degree of destruction parameter such as $1 \leq |\mathcal{H}^R| \leq \xi \cdot |\mathcal{H}|$.

$\mathcal{C}^R$ is defined as the set of all car travels in $\mathcal{H}^R$ and $\mathcal{D}^R$ is defined as the set of duties in solution that contain removed car travels, i.e. $\mathcal{D}^R = \{d \mid g_{id} = 1, i \in \mathcal{C}^R, d \in \mathcal{D}\}$.

The removed duties, car travels and car matches, $\mathcal{D}^R, \mathcal{C}^R$ and $\mathcal{H}^R$, are added to $\mathcal{D}', \mathcal{C}'$ and $\mathcal{H}'$ respectively.

A.4.2.2 Repair Methods

The repair methods of the ALNS are mainly intended to be fast heuristics; simple greedy insertion and regret heuristics have regularly been applied as repair methods (Pisinger and Ropke, 2007 and Lusby et al., 2016). The DSPSC is a very tightly constrained problem where exploring a neighborhood for a feasible solution could be a challenge. To tackle the challenge of finding feasible solutions for the lot sizing problem, Muller et al. (2012) used a MIP solver for repairing solution. Similarly, we use a MIP solver (ILOG CPLEX) as part of the repair phase of the heuristic. Muller et al. (2012) define two MIP based repair methods and differentiate them by either fixing or bounding the variables based on their values in the destroyed solution. However, in our approach, we differentiate the repair methods by the neighborhood defined for the MIP solver. The variables in the destroyed solution, given by $\hat{D}, \hat{C}$ and $\hat{H}$, are used as a starting solution for the MIP solver but are set as “free” variables. The repair methods reduce the search space by defining the neighborhoods, $\mathcal{D}^N, \mathcal{C}^N$ and $\mathcal{H}^N$, for the MIP solver which is capable of exploring numerous solutions within the defined neighborhood. Consequently, the repair methods are evaluated based on the quality of the constructed neighborhoods.

By solving a restricted subproblem, the MIP solver helps the local search to move from the current solution to a neighboring solution and the new improved
solution determines the neighborhood that will be defined by the local search. Limiting the number of branch nodes to be explored or the time permitted for the MIP solver were suggested by Muller et al. (2012) in order to speed up the solution process. We keep a time limit, $n_{time}$, and the solution generated by the MIP solver is used to evaluate the repair methods.

After a solution has been destroyed, there are unmatched car travels, $C^U$, and uncovered bus trips, $T^U = \{ t \mid \sum_{d \in D} a_{td} = 0, \ t \in T \}$ in the solution. Hence, we define a repair method that constructs a neighborhood to focus primarily on covering the uncovered bus trips in the solution and another repair method that focuses on matching the unmatched car travels in the solution. The descriptions of the repair methods are as follows,

1. Neighborhood defined by duties that cover the uncovered bus trips
   Given a set of uncovered bus trips in the solution, $T^U$, the neighborhood is formally defined as the set of duties that cover at least one of the trips as shown in Equation (A.17).
   \[ D^N = \{ d \mid \sum_{t \in T^U} a_{td} \geq 1, \ d \in D^' \} \quad (A.17) \]
   Depending on the duties in the neighborhood, the set of car travels in the neighborhood is defined as $C^N = \{ i \mid g_{id} = 1, \ i \in C^', \ d \in D^N \}$ and subsequently the neighborhood car matches as $H^N = \{ (i, j) \mid (i, j) \in H^', \ i \in C^N, \ j \in C^N \}$.

2. Neighborhood defined by duties that match with the unmatched car travels
   Given a set of unmatched car travels in the solution, $C^U$, we denote $C^M$ as the set of car travels from $C^'$ that can match with the unmatched car travels. For instance, car travel $j \in C^'$ can match with unmatched car travel $i \in \tilde{C}$, i.e $(i, j) \in H^'$. Consequently, the set is defined as $C^M = \{ j \mid (i, j) \in H^', \ i \in \tilde{C}, \ j \in C^' \}$. Therefore, the neighborhood is defined as the set of duties that contain one or more car travels that can match with the unmatched car travels in the solution as shown in Equation (A.18). The car travels and car matches in the neighborhood are defined based on $D^N$ in similar manner as the previously described repair neighborhood.
   \[ D^N = \{ d \mid g_{id} = 1, \ d \in D^', \ i \in C^M \} \quad (A.18) \]

The size of the neighborhood, $D^N$, depends on the impact of the destroy methods and in most instances, destroying even a small fraction of the current solution creates a large neighborhood. For the DSP, Lourenço et al. (2001) considered
a candidate list strategy where duties were evaluated based on a penalized cost and only duties with a cost less than or equal to the average cost were inserted in the candidate list. Similarly, we create a duty candidate list, where only the best \(d\) duties in terms of \(\Delta_d\), where \(d \in D^N\), are considered. The candidate list makes the subproblem tractable for the MIP solver and provides solutions in quick computation time. The size of \(H^N\) could also potentially be quite large and hence the size of the neighborhood is controlled by carIdleTime() of car matches. For example, the staff car match candidate list only considers matches that are less than a maximum car idle time \((\eta_{\text{car}})\) of 120 minutes, i.e. carIdleTime\((i, j)\) \(\leq 120\) where \((i, j) \in H^N\). The matheuristic procedure is shown in Algorithm 7.

Algorithm 7: Matheuristic

1. **Initialization:** \(s \leftarrow \text{GreedyAlgorithm}(), s^* \leftarrow s\);
2. \(\theta \leftarrow \text{CalculateInitialTemperature}(s, \omega)\);
3. \(\rho \leftarrow \text{InitializeMethodWeights}()\);
4. \(\omega \leftarrow \text{InitializeMethodScores}()\);
5. \(\nu \leftarrow \text{InitializeMethodAttempts}()\);
6. for \(\kappa \leftarrow 1, n_{\text{seg}}\) do
   7.   for \(\eta \leftarrow 1, n_{\text{iter}}\) do
      8.     Select destroy and repair methods \(\mu \in \tau^-\) and \(\gamma \in \tau^+\) using \(\rho\);
      9.     \(s' \leftarrow \text{Destroy}(s, \mu, \xi, B)\);
     10.    \(s' \leftarrow \text{Repair}(s', \gamma, \eta_{\text{duty}}, \eta_{\text{car}}, n_{\text{time}})\); // Solve using MIP solver
     11.    \(\delta \leftarrow f(s') - f(s)\);
     12.    if \(\delta < 0\) or \(\exp \frac{-\delta}{\theta} > \text{random}[0, 1]\) then
          13.       \(s \leftarrow s'\);
     14.    end
     15.    if \(f(s') < f(s*)\) then
          16.       \(s^* \leftarrow s'\);
     17.    end
     18.    \(\omega \leftarrow \text{UpdateMethodScores}(\psi)\);
     19.    \(\nu \leftarrow \text{UpdateMethodAttempts}()\);
   10.   end
     11. end
     12. \(\rho \leftarrow \text{UpdateMethodWeights}(\omega, \nu, \lambda)\);
     13. \(\omega \leftarrow \text{ResetMethodScores}()\);
     14. \(\nu \leftarrow \text{ResetMethodAttempts}()\);
     15. \(\theta \leftarrow \theta \cdot \alpha\);
   9. end
26. return \(s^*\)
A.5 Computational Study

A.5.1 Instances

Table A.2 provides an overview of the test instances obtained from a Danish and a Swedish transport company. \textit{SE1\_OP} represents the instances from the Swedish company and the instances from the Danish company are represented by \textit{DK1\_OP} and \textit{DK2\_OP}. Instances \textit{SE1\_OP5}, \textit{DK1\_OP6} and \textit{DK2\_OP9} are known to be the complete instances that were used to extract other instances. The three complete instances are highlighted in light gray in Table A.2. The instances were categorized into small, medium and large sized instances so that the matheuristic could be tested for a wide range of instances. Some of the instances are larger than what one would find currently in the literature. The instances are available at http://doi.org/10.5281/zenodo.1442661.

| Category | Instance | \(|T|\) | \(|D|\) | \(|C|\) | \(|H|\) | \(Q\) |
|----------|----------|--------|--------|--------|--------|------|
| Small    | \textit{SE1\_OP1} | 44     | 1239   | 91     | 345    | 4    |
|          | \textit{SE1\_OP2} | 39     | 8880   | 100    | 391    | 1    |
|          | \textit{DK1\_OP1} | 23     | 754    | 65     | 148    | 1    |
|          | \textit{DK1\_OP2} | 73     | 1789   | 140    | 781    | 1    |
|          | \textit{DK2\_OP1} | 84     | 7660   | 149    | 2621   | 2    |
|          | \textit{DK2\_OP2} | 96     | 18370  | 280    | 4134   | 1    |
| Medium   | \textit{SE1\_OP3} | 131    | 39683  | 294    | 3081   | 4    |
|          | \textit{DK1\_OP3} | 152    | 41908  | 302    | 8176   | 2    |
|          | \textit{SE1\_OP4} | 217    | 193652 | 501    | 8372   | 5    |
|          | \textit{DK1\_OP4} | 279    | 195972 | 710    | 17744  | 4    |
|          | \textit{DK2\_OP3} | 305    | 86703  | 753    | 20330  | 3    |
| Large    | \textit{SE1\_OP5} | 293    | 621508 | 731    | 12293  | 6    |
|          | \textit{DK1\_OP5} | 384    | 686499 | 1149   | 34558  | 5    |
|          | \textit{DK2\_OP4} | 649    | 511803 | 1514   | 64238  | 4    |
|          | \textit{DK2\_OP5} | 840    | 686370 | 1862   | 77778  | 5    |
|          | \textit{DK2\_OP6} | 924    | 752705 | 2141   | 91371  | 6    |
|          | \textit{DK1\_OP6} | 571    | 1205058| 1746   | 50023  | 6    |
|          | \textit{DK2\_OP7} | 1211   | 1015011| 2852   | 150205 | 8    |
|          | \textit{DK2\_OP8} | 1414   | 1187194| 3512   | 189671 | 12   |
|          | \textit{DK2\_OP9} | 1769   | 1738055| 4560   | 267506 | 16   |
|          | \textit{DK2\_OP10} | 1769   | 1738055| 4560   | 267506 | 15   |

Table A.2: Size of test instances. \(|T|\) represents the number of bus trips, \(|D|\) represents the number of duties generated, \(|C|\) represents the number of number of car travels, \(|H|\) represents the number of car matches and \(Q\) represents the number of staff cars at the depot.

Table A.2 shows that all instances involve the use of staff cars. In the DSP, travels by foot for short distances and bus travels are commonly allowed. A bus travel usually occurs when the driver is a passenger on another bus to reach a
A Matheuristic for the Driver Scheduling Problem with Staff Cars
designated bus stop and the set covering constraints (A.3) allow drivers to use bus travels. One could argue that staff cars may not be needed if bus stops could be reached by bus or by foot. To analyze the importance of staff cars, the small and medium sized instances are tested with and without car travels. It was found that all bus trips could be covered when staff cars are put into use. However, when the instances do not involve car travels, it was found that, on an average, 41% of the bus trips could not be covered. The labor rules such as the minimum number of breaks and maximum time between breaks highly influence the feasibility of driver duties. Moreover, due to limitations in the city network where breaks are allowed only at a few bus stops, drivers have to travel between bus stops to have sufficient breaks during the day. Car travel is the most suitable option that allows drivers to reach bus stops in a timely manner and it is, hence, argued that staff cars are often necessary to generate feasible driver duties.

The mathematical formulation (A.2) - (A.12) solves the DSPSC using an integrated approach where the drivers and the staff cars are scheduled simultaneously. Another method of solving the DSPSC is by a sequential approach, where the DSP is solved first and independent of the staff car problem. After solving the DSP, car travels in the final set of duties are chosen as the input for the staff car problem, which is concerned with finding a feasible set of car matches that respect the number of staff cars available at the depot. The small and medium sized instances are solved by integrated and sequential approaches using a MIP solver. Table A.3 compares the two approaches and the results show that the sequential approach is superior to the integrated approach in terms of total paid time for drivers, where the average improvement for small and medium sized instances are 5.31% and 2.13% respectively. However, the sequential approach often leads to infeasible solutions with the medium sized instances having, on an average, 10 unmatched car travels out of 47. The integrated approach provides feasible solutions; however, the computation time required to solve the DSPSC is significantly larger than that of the sequential approach. For example, the integrated approach did not find the optimal solution in 10 hours for the medium sized instance $DK2\_OP3$, whereas only 18 seconds were required by the sequential approach. This computational study shows that simultaneously scheduling the drivers and the staff cars is a highly complex problem that often requires long computation times. Due to the computational advantage of the sequential approach, it can be considered for solving the DSPSC with alternative services. For example, taxis can be used by transport companies to fulfill the unmatched car travels in the solution. Transport companies would have to consider the commercial viability of using taxis to fulfill such car travels without losing much of the 2.13% savings made by the sequential approach for medium sized instances. However, the vehicle policies of companies we work with do not allow for any outsourced services and the drivers are required to use the staff cars. Hence, the DSPSC only considers solving the problem with a
given number of staff cars at the depot and the work carried out in this paper does not consider alternative services.

<table>
<thead>
<tr>
<th>Category</th>
<th>Instance</th>
<th>Integrated</th>
<th>Sequential</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>solution</td>
<td>gap(%)</td>
</tr>
<tr>
<td>Small</td>
<td>SE1_OP1</td>
<td>4883</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>SE1_OP2</td>
<td>3144</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>DK1_OP1</td>
<td>1914</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>DK2_OP1</td>
<td>3120</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>DK1_OP2</td>
<td>5867</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>DK2_OP2</td>
<td>2795</td>
<td>0.00</td>
</tr>
<tr>
<td>Medium</td>
<td>SE1_OP3</td>
<td>10925</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>DK1_OP3</td>
<td>10927</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>DK1_OP4</td>
<td>3120</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>DK2_OP3</td>
<td>2795</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table A.3: Comparison between integrated and sequential approaches. \(|\bar{C}|\) represents the number of car travels in the final solutions and \(|C^U|\) represents the number of unmatched car travels in the solution.

A.5.2 Parameter setup

The number of segments, \(n_{\text{seg}}\), is set to 50 and the number of iterations to be performed in each segment, \(n_{\text{iter}}\), is 15. For the SA accept criterion, \(\theta_{\text{start}}\) is calculated such that a solution 5% (\(\omega\)) worse than \(f(s)\) is accepted with probability 0.5, i.e \(\theta_{\text{start}} = -f(s) \times 0.05 \log 0.5\), and \(\alpha\) is set to 0.8. Table A.4 shows the results for different values of \(\alpha\). The average number of solutions accepted in the SA framework decreases as \(\alpha\) is decreased. The solution quality is determined by calculating the average gap between the solutions obtained from the matheuristic and the best known solution obtained from the MIP solver.

<table>
<thead>
<tr>
<th>(\alpha)</th>
<th>0.99</th>
<th>0.9</th>
<th>0.8</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. accepted solutions</td>
<td>107.8</td>
<td>67.4</td>
<td>40</td>
<td>23.8</td>
</tr>
<tr>
<td>Avg. gap(%)</td>
<td>-1.03</td>
<td>-2.23</td>
<td>-2.51</td>
<td>-2.02</td>
</tr>
</tbody>
</table>

Table A.4: Test results for parameter \(\alpha\).

The score parameters of the matheuristic are \(\psi_1 = 25\), \(\psi_2 = 15\), \(\psi_3 = 5\) and \(\psi_4 = 0\), and the reaction factor \(\lambda\) is set to 0.1. For the destroy methods, the degree of destruction parameter, \(\xi\), is set to 0.2, 0.1 and 0.025 for small, medium and large instances respectively. The degree of randomization \(B\) for the worst removal is set to 4. The time limit \(n_{\text{time}}\) of the MIP solver in the repair methods is set to 0.5, 2 and 3 seconds for small, medium and large instances respectively. For setting \(\eta_{\text{duty}}\) and \(\eta_{\text{car}}\), tests were performed on a
Danish (DK2_OP9) and a Swedish instance (SE1_OP5). Table A.5 shows that the size of the neighborhood varies depending on the problem instance and on the applied repair method. Parameter $\eta_{duty}$ was tested with different values as shown in Table A.6, where it can be seen that the average size of $H^N$ increases as $\eta_{duty}$ is increased. The chosen value for parameter $\eta_{duty}$ is 6000 and parameter $\eta_{car}$ is adapted based on the size of $H^N$. If $|H^N| \geq 14000$, which is the approximate average from Table A.6, then $\eta_{car}$ is set to 120 else it is set to 180.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Repair method 1</th>
<th>Repair method 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>DK2_OP9</td>
<td>117227.67</td>
<td>149249.63</td>
</tr>
<tr>
<td>SE1_OP5</td>
<td>40072.08</td>
<td>31851.04</td>
</tr>
</tbody>
</table>

Table A.5: Average size of $D^N$ defined by the repair methods.

<table>
<thead>
<tr>
<th>Instance</th>
<th>$\eta_{duty}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5000</td>
</tr>
<tr>
<td>DK2_OP9</td>
<td>20636.87</td>
</tr>
<tr>
<td>SE1_OP5</td>
<td>5538.86</td>
</tr>
</tbody>
</table>

Table A.6: Average size of $H^N$ for different values of parameter $\eta_{duty}$.

### A.5.3 Performance of destroy and repair methods

Instances SE1_OP5 and DK2_OP9 are tested with different combinations of destroy and repair methods of the matheuristic as shown in Table A.7. Each combination or strategy is tested 5 times on the instances and Table A.7 reports the average of the 5 runs. An observation made from the study is that strategies involving repair method 2 comparatively provide weaker solutions and, in some cases, do not yield feasible solutions within the iteration limit of the matheuristic. Repair method 1 consistently performs well when combined with the destroy methods. Figures A.4 and A.5 show an example of how the weights of the repair and destroy methods progressed during the execution of the matheuristic for the instances from the Swedish and Danish transport companies. The figures illustrate that the neighborhood based on the uncovered bus trips (repair method 1) outperforms the neighborhood based on unmatched car travels (repair method 2). For DK2_OP9 instance, it was observed that strategies with repair method 2 often created a large neighborhood that increased the computation time required for defining the duty candidate list $\eta_{duty}$. Hence, the computation time for strategies with repair method 2 was, on average, 1.5 times longer than that of the strategies with repair method 1. Moreover, since repair method 2 does not consider the uncovered bus trips in the destroyed solution, it appears to define an ineffective neighborhood. Repair method 2 was initially developed
for diversifying the search space; however, the results clearly indicate that the method does not aid the matheuristic much in improving the solution quality. Thus, it is decided to remove repair method 2 from the matheuristic setup.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Strategy</th>
<th>Avg. gap (%)</th>
<th>Avg. time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE1_OP5</td>
<td>Destroy method 1, Repair method 1</td>
<td>5.35</td>
<td>2355.27</td>
</tr>
<tr>
<td></td>
<td>Destroy method 2, Repair method 1</td>
<td>5.89</td>
<td>1728.54</td>
</tr>
<tr>
<td></td>
<td>Destroy method 3, Repair method 1</td>
<td>4.64</td>
<td>3350.43</td>
</tr>
<tr>
<td></td>
<td>All destroy methods, Repair method 1</td>
<td>3.69</td>
<td>2245.61</td>
</tr>
<tr>
<td></td>
<td>Destroy method 1, Repair method 2</td>
<td>inf</td>
<td>2567.84</td>
</tr>
<tr>
<td></td>
<td>Destroy method 2, Repair method 2</td>
<td>inf</td>
<td>2933.47</td>
</tr>
<tr>
<td></td>
<td>Destroy method 3, Repair method 2</td>
<td>15.09</td>
<td>1452.27</td>
</tr>
<tr>
<td></td>
<td>All destroy methods, Repair method 2</td>
<td>inf</td>
<td>2312.02</td>
</tr>
<tr>
<td></td>
<td>All destroy methods, all repair methods</td>
<td>4.26</td>
<td>2192.77</td>
</tr>
<tr>
<td>DK2_OP9</td>
<td>Destroy method 1, Repair method 1</td>
<td>-1.21</td>
<td>4879.65</td>
</tr>
<tr>
<td></td>
<td>Destroy method 2, Repair method 1</td>
<td>-1.8</td>
<td>3962.99</td>
</tr>
<tr>
<td></td>
<td>Destroy method 3, Repair method 1</td>
<td>-1.78</td>
<td>4213.43</td>
</tr>
<tr>
<td></td>
<td>All destroy methods, Repair method 1</td>
<td>-2.69</td>
<td>4435.53</td>
</tr>
<tr>
<td></td>
<td>Destroy method 1, Repair method 2</td>
<td>6.41</td>
<td>7070.09</td>
</tr>
<tr>
<td></td>
<td>Destroy method 2, Repair method 2</td>
<td>8.23</td>
<td>7933.52</td>
</tr>
<tr>
<td></td>
<td>Destroy method 3, Repair method 2</td>
<td>4.08</td>
<td>6180.8</td>
</tr>
<tr>
<td></td>
<td>All destroy methods, Repair method 2</td>
<td>3.56</td>
<td>6567.9</td>
</tr>
<tr>
<td></td>
<td>All destroy methods, all repair methods</td>
<td>-1.67</td>
<td>5078.36</td>
</tr>
</tbody>
</table>

Table A.7: Performance of destroy and repair methods. The results are based on an average of 5 runs and 'inf' indicates that a feasible solution could not be found within the iteration limit of the matheuristic in any one of the runs.

A.5.4 Results

The solutions obtained from the MIP solver (ILOG CPLEX 12.8) are used as benchmarks to evaluate the performance of the matheuristic. The instances are solved by the MIP solver on an Intel Xeon E5-2680 v2 @ 2.80GHz with 128 GB memory and the results from using single thread and multi threads (10 threads) are reported in Table A.8. The time limit of the MIP solver is 36,000 seconds (10 hours). Due to practical limitations of extracting data from TGE’s integrated system, the matheuristic could not be tested on the machine that was used to provide the benchmark solutions. Table A.9 shows the summary of the results from the matheuristic that was tested on an Intel core i5-5287U @ 2.9 GHz machine with 16 GB memory. The matheuristic is implemented in Java and has been run 10 times for each instance and the results are calculated as the average of the 10 runs.

For single thread applications, the processor used to test the matheuristic is comparable to the processor used to run the MIP solver in terms of speed. However, the overall CPU benchmarks reveal that the processor used for the
MIP solver is approximately 3.4 times faster than the processor used to test the matheuristic\(^3\). The results in Table A.8 and Table A.9 are reported as observed. The best solution obtained from running the MIP solver single and multi threaded is used to evaluate the performance of the matheuristic. In most cases, the MIP solver provided best solutions in the multi thread environment. For two instances (\textit{DK2\_OP6} and \textit{DK2\_OP7}), the best solutions are obtained rating of 1810, whereas the single thread rating of the processor used to test the matheuristic is 1847.

\(^3\)The overall CPU rating of the processor used for the MIP solver is 15752 while that of the matheuristic is 4681.
in the single thread environment.

The small instances are solved to optimality with ease by the MIP solver. The matheuristic achieves comparable results for all small instances except one (DK2_OP2), which has an average gap of 2.33% from the optimal solution. For medium sized instances, the MIP solver fails to prove optimality for two instances (SE1_OP4 and DK2_OP3) within the time limit (10 hours); however, the integrality gap is very small (<1%). The matheuristic provides solutions less than 2.5% from optimality for instances with optimal solutions. For the two instances that could not be solved to optimality, the gaps are found to be 3.72% and 5.93% respectively. For large instances, the MIP solver could prove
optimal for only one instance (DK1_OP5) and the integrality gap for large instances of DK2_OP (DK2_OP7 to DK2_OP10) is quite large, an average gap of 14.89%. The matheuristic finds improved solutions for 4 out of the 10 large instances and the improvement is found to be 0.28 to 6.95% on average. Moreover, the time taken to obtain these solutions are less than 80 minutes. One of the major drawbacks of the heuristic is that it does not provide any lower bound (LB) information that could be used to evaluate the quality of the improved solutions. However, by considering the LB information provided by the MIP solver, it is estimated that the 4 improved instances are around 11 to 12.77% from the optimal solution. Table A.9 also shows the average time taken by the matheuristic to find the first feasible solution. With the aid of the greedy heuristic (Algorithm 5), feasible solutions for large instances are found in the range of 37 seconds to 14 minutes.

<table>
<thead>
<tr>
<th>Category</th>
<th>Instance</th>
<th>MIP (single thread)</th>
<th>MIP (multi thread)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>solution</td>
<td>gap(%)</td>
</tr>
<tr>
<td>Small</td>
<td>SE1_OP1</td>
<td>4883</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>SE1_OP2</td>
<td>3144</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>DK1_OP1</td>
<td>1914</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>DK1_OP2</td>
<td>3129</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>DK1_OP3</td>
<td>5867</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>DK2_OP1</td>
<td>2795</td>
<td>0.00</td>
</tr>
<tr>
<td>Medium</td>
<td>SE1_OP3</td>
<td>10925</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>DK1_OP3</td>
<td>10927</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>SE1_OP4</td>
<td>17765</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>DK1_OP4</td>
<td>20253</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>DK2_OP3</td>
<td>12650</td>
<td>2.83</td>
</tr>
<tr>
<td>Large</td>
<td>SE1_OP5</td>
<td>23833</td>
<td>2.01</td>
</tr>
<tr>
<td></td>
<td>DK1_OP5</td>
<td>27773</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>DK2_OP4</td>
<td>26606</td>
<td>14.71</td>
</tr>
<tr>
<td></td>
<td>DK2_OP5</td>
<td>36568</td>
<td>15.4</td>
</tr>
<tr>
<td></td>
<td>DK2_OP6</td>
<td>39081</td>
<td>10.72</td>
</tr>
<tr>
<td></td>
<td>DK1_OP6</td>
<td>42713</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td>DK2_OP7</td>
<td>57507</td>
<td>11.27</td>
</tr>
<tr>
<td></td>
<td>DK2_OP8</td>
<td>76733</td>
<td>16.61</td>
</tr>
<tr>
<td></td>
<td>DK2_OP9</td>
<td>96003</td>
<td>21.14</td>
</tr>
<tr>
<td></td>
<td>DK2_OP10</td>
<td>97374</td>
<td>22.16</td>
</tr>
</tbody>
</table>

Table A.8: Results from the MIP solver.

Since the processors used for testing the two methods are different and their computation times vary, it is difficult to directly compare their performances. Hence, the MIP solver is tested with a time limit of 2 hours, which is comparable to the computation times of the matheuristic and Table A.10 compares the results of the matheuristic to that of the MIP solver. The matheuristic outperforms the MIP solver for 6 out of the 10 large instances and the improvement is found to be 7 to 15% on average.
### Table A.9: Results from the matheuristic.

<table>
<thead>
<tr>
<th>Category</th>
<th>Instance</th>
<th>Gap (%)</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>best</td>
<td>worst</td>
</tr>
<tr>
<td>Small</td>
<td>SE1_OP1</td>
<td>0.00</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>SE1_OP2</td>
<td>0.00</td>
<td>3.63</td>
</tr>
<tr>
<td></td>
<td>DK1_OP1</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>DK2_OP1</td>
<td>0.00</td>
<td>4.26</td>
</tr>
<tr>
<td></td>
<td>DK1_OP2</td>
<td>0.00</td>
<td>0.55</td>
</tr>
<tr>
<td></td>
<td>DK2_OP2</td>
<td>0.11</td>
<td>6.69</td>
</tr>
<tr>
<td>Medium</td>
<td>SE1_OP3</td>
<td>0.61</td>
<td>3.73</td>
</tr>
<tr>
<td></td>
<td>DK1_OP3</td>
<td>0.66</td>
<td>3.39</td>
</tr>
<tr>
<td></td>
<td>SE1_OP4</td>
<td>2.86</td>
<td>4.85</td>
</tr>
<tr>
<td></td>
<td>DK1_OP4</td>
<td>0.78</td>
<td>1.68</td>
</tr>
<tr>
<td></td>
<td>DK2_OP3</td>
<td>3.87</td>
<td>9.04</td>
</tr>
<tr>
<td>Large</td>
<td>SE1_OP5</td>
<td>2.72</td>
<td>4.51</td>
</tr>
<tr>
<td></td>
<td>DK1_OP5</td>
<td>3.91</td>
<td>5.53</td>
</tr>
<tr>
<td></td>
<td>DK2_OP4</td>
<td>1.08</td>
<td>5.23</td>
</tr>
<tr>
<td></td>
<td>DK2_OP5</td>
<td>-1.92</td>
<td>2.49</td>
</tr>
<tr>
<td></td>
<td>DK2_OP6</td>
<td>-0.09</td>
<td>2.11</td>
</tr>
<tr>
<td></td>
<td>DK1_OP6</td>
<td>3.53</td>
<td>5.86</td>
</tr>
<tr>
<td></td>
<td>DK2_OP7</td>
<td>-0.76</td>
<td>0.57</td>
</tr>
<tr>
<td></td>
<td>DK2_OP8</td>
<td>-4.26</td>
<td>-3.2</td>
</tr>
<tr>
<td></td>
<td>DK2_OP9</td>
<td>-3.48</td>
<td>-1.63</td>
</tr>
<tr>
<td></td>
<td>DK2_OP10</td>
<td>-8.17</td>
<td>-5.68</td>
</tr>
</tbody>
</table>

### A.5.5 Sensitivity analysis

The matheuristic involves 14 parameters and finding the optimal values of parameters for each instance is a very tedious and time consuming process. In this paper, we chose a common set of parameter values for each category that was based on the sizes of the instances. However, it is believed that the structure and the characteristics of the instances also have to be considered when tuning the parameters. For the largest category, Swedish (SE1_OP5) and Danish (DK2_OP9) instances were taken as the training instances and parameter $\eta_{\text{car}}$ was adapted such that if $|\mathcal{H}^N| \geq 14000$ then it was set to 120 otherwise it was set to 180. Hence, the matheuristic is over-fitted for the aforementioned instances and is potentially prone to a large deviation in performance for an unseen test instance, which may possess different characteristics. To analyze the sensitivity of the matheuristic, we tested DK2_OP9 instance with different threshold values of $|\mathcal{H}^N|$ and values of $\eta_{\text{car}}$ as shown in Table A.11. The best known solution provided by the MIP solver is used to calculate the average gap (%) of solutions yielded by different settings. The results show that the performance of the matheuristic deteriorates with increase in threshold value of $|\mathcal{H}^N|$ and value of $\eta_{\text{car}}$, which indicate that the matheuristic is sensitive to parameter values.
### Table A.10: Comparison of results from the MIP solver and results from the matheuristic.

<table>
<thead>
<tr>
<th>Category</th>
<th>Instance</th>
<th>MIP (multi thread)</th>
<th>Matheuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium</td>
<td>SE1_OP3</td>
<td>10925 0.00 130.31</td>
<td>2.33 883.29</td>
</tr>
<tr>
<td></td>
<td>DK1_OP3</td>
<td>10927 0.00 48.21</td>
<td>1.93 1715.3</td>
</tr>
<tr>
<td></td>
<td>SE1_OP4</td>
<td>17836 1.27 7200.4</td>
<td>3.31 2576.29</td>
</tr>
<tr>
<td></td>
<td>DK1_OP4</td>
<td>20253 0.00 1002.92</td>
<td>1.25 1708.71</td>
</tr>
<tr>
<td></td>
<td>DK2_OP3</td>
<td>12554 1.99 7200.34</td>
<td>5.19 2092.41</td>
</tr>
<tr>
<td>Large</td>
<td>SE1_OP5</td>
<td>23631 1.18 7201.98</td>
<td>3.36 2383.63</td>
</tr>
<tr>
<td></td>
<td>DK1_OP5</td>
<td>27773 0.00 3687.24</td>
<td>4.48 1870.26</td>
</tr>
<tr>
<td></td>
<td>DK2_OP5</td>
<td>37651 17.95 7203.34</td>
<td>-6.66 2704.05</td>
</tr>
<tr>
<td></td>
<td>DK2_OP6</td>
<td>43492 12.57 7203.33</td>
<td>-8.99 3414.22</td>
</tr>
<tr>
<td></td>
<td>DK1_OP6</td>
<td>42394 1.07 7204.23</td>
<td>4.21 2410.03</td>
</tr>
<tr>
<td></td>
<td>DK2_OP7</td>
<td>65658 22.27 7205.68</td>
<td>-12.66 3389.88</td>
</tr>
<tr>
<td></td>
<td>DK2_OP8</td>
<td>83117 22.81 7206.42</td>
<td>-14.07 4197.76</td>
</tr>
<tr>
<td></td>
<td>DK2_OP9</td>
<td>102831 26.88 7215.21</td>
<td>-14.74 4700.99</td>
</tr>
<tr>
<td></td>
<td>DK2_OP10</td>
<td>102844 26.92 7213.6</td>
<td>-14.72 4726.6</td>
</tr>
</tbody>
</table>

### Table A.11: Sensitivity analysis of the matheuristic for different threshold values of $|\mathcal{H}_N|$ and values of $\eta_{car}$.

| $|\mathcal{H}_N|$ | $\eta_{car}$ | 60  | 90  | 120 | 150 | 180 |
|-----------------|--------------|-----|-----|-----|-----|-----|
| 10000           | -2.34 -2.7  -1.83 -1.85 | -1.51 -1.01 |
| 12000           | -2.74 -2.77 -2.04 -1.45 | -1.16 -1.02 |
| 14000           | -2.73 -2.71 -2.52 -1.42 | -1.4  0.43 |
| 16000           | -2.36 -2.21 -1.36 -1.29 | 0.63  1.59 |
| 18000           | -1.44 -1.61 -1.03 2.29  | 3.75  3.8 |
| 20000           | -                |     |     |     |     |

### A.6 Conclusion

In this paper, we have introduced the DSPSC and presented a matheuristic to solve the problem. Computational study with real-life instances from Denmark and Sweden revealed that small and medium sized instances were solved with ease by the MIP solver. However, for larger instances with more than 6 cars and 1200 bus trips, the integrality gap on average was around 14.89%. The matheuristic provided better solutions, with comparable computation times, for 6 out of the 10 large instances. On larger instances, the improvement is approximately 13-15% on average.

Therefore, in most cases, the proposed method is superior than an approach based on solving the problem as a MIP problem for large instances in terms of solution quality and computation time. However, integrating the matheuristic as part of a decision support tool could be a challenging task. For solving the
DSP, other practical conditions may exist such as maximum number of duties, maximum/minimum average working time of the duties and occasionally the objective is to minimize the total number of duties rather than minimizing the cost. Hence, in addition to solution quality and computation time, the transport industry demands a flexible decision support tool that allows for analyzing various scenarios, which will be beneficial during the planning process. Consequently, the devised matheuristic should have the ability to adapt to the diverse requirements from the users of the decision support tool. Since the users of the tool generally have limited knowledge of OR, user-friendliness is considered to be another key factor for successful integration of heuristics into decision support tools. The work carried out in this paper aimed at testing the matheuristic for a wide variety of problems from Danish and Swedish transport companies and creating a set of parameter values for each category. However, if a new set of problems with varying sizes is given, it may possess different characteristics. Parameter tuning is considered to be a time consuming and tedious process, and approaches such as F-Race (Birattari et al., 2010) have been addressed in the literature for automatic parameter configuration. Furthermore, problem-dependent knowledge may still be needed to make the heuristic effective and perform consistently, which requires highly skilled practitioners. In conclusion, the need to design flexible and user-friendly heuristics is considered as a primary challenge for real-life implementation and could, hence, be seen as future areas of research.

The DSPSC is a practical problem with many variations and we hope to inspire other researchers in the area of vehicle and driver scheduling. One interesting variation of the problem, which is of significant importance to the transport industry, is a car routing problem that could have cars visiting multiple nodes, rather than a single node, before returning to the depot.


A Column Generation Approach for the Driver Scheduling Problem with Staff Cars

Shyam S.G. Perumal\textsuperscript{a,b}, Jesper Larsen\textsuperscript{a}, Richard M. Lusby\textsuperscript{a}, Morten Riis\textsuperscript{b}, Tue R.L. Christensen\textsuperscript{b}

\textsuperscript{a}Department of Technology, Management and Economics, Technical University of Denmark, Kgs. Lyngby, Denmark
\textsuperscript{b}QAMPO ApS, Aarhus, Denmark

\textsuperscript{1}Abstract: Given a set of timetabled bus trips, transport companies are faced with the challenge of finding a feasible driver schedule that covers all trips and abides by various labor union regulations. The regulations are primarily concerned with providing sufficient breaks for the drivers during the day. Practical limitations in the city network enforce drivers to travel by cars between bus stops to have breaks.

\textsuperscript{1}Submitted to Journal of Scheduling, March 2019
Transport companies have a limited number of cars, known as staff cars, which have to be returned to their respective depots at the end of the day. The simultaneous scheduling of drivers and staff cars for the drivers is known as the driver scheduling problem with staff cars (DSPSC). It is estimated that the DSPSC accounts for 60% of a bus company's operational expense, and this paper proposes a column generation approach that attempts to minimize operational expense. The column generation framework iterates between a master problem, a subproblem for generating driver variables and a subproblem for generating staff car variables. The subproblem related to the drivers is formulated as a resource constrained shortest path problem, which is solved by a dynamic programming approach. Several heuristic branching strategies are explored to find integer solutions. The proposed methodology is tested on eight real-life instances from seven Northern European bus companies. A comparison with a state-of-the-art mixed integer programming (MIP) solver and an adaptive large neighborhood search (ALNS) heuristic indicate that the column generation approach provides improved solutions for six instances and the average improvement is 1.45%.

**Keywords:** Transportation, Driver scheduling problem, Heuristics, Column generation

### B.1 Introduction

Since the 1980s, researchers from the operations research (OR) community have been working alongside practitioners from the public transport industry to solve scheduling and routing problems. Some of the early applications of OR techniques within the public bus transport industry include Smith and Wren (1988) and Desrochers and Soumis (1989), where the authors attempted to solve the driver scheduling problem (DSP). Given a set of timetabled bus trips for a day, the DSP is concerned with finding a driver schedule that covers all bus trips with minimal operational expense. The feasibility of a driver schedule is influenced by numerous rules and regulations imposed by governments and labor unions. Additionally, practical conditions within cities challenge transport companies to construct efficient driver schedules. With ever changing policies and business needs, transport companies constantly seek techniques that can aid them in managing various complexities and improve efficiency of their operations. Computer hardware and commercial state-of-the-art mixed integer programming (MIP) solvers such as CPLEX have improved tremendously over the last decade. Modern MIP technology has become very effective in solving practical
B.2 Related Literature

problems and has, thus, made its way into various commercial applications. For more information on the progress of commercial MIP solvers refer to Bixby and Rothberg (2007) and Achterberg and Wunderling (2013). Trapeze Group Europe (TGE) is an international provider of decision support tools within planning and operations for both public and private transport companies. QAMPO is a developer of algorithms based on OR techniques that solve scheduling problems across various industries. The partnership between TGE and QAMPO strives to fulfill business needs of transport companies with the use of modern MIP technology. However, due to the enormous sizes and complexities of most real life cases, the DSP cannot explicitly be tackled by state-of-the-art MIP solvers. Hence, in practice, one may resort to near-optimal solutions. Thus, areas within public transport and OR have to be researched and investigated for developing techniques that can more efficiently solve the DSP.

The driver scheduling problem with staff cars (DSPSC) was first introduced by Perumal et al. (2019), which describes the use of cars by the bus drivers as part of the travel activities during the day. The problem is restricted by the number of cars available at a transport company’s depot and the cars have to be returned to the depot at the end of the day. The DSPSC adds further complexity to the NP-hard DSP (Fischetti et al., 1987) and is a highly relevant problem in the public transport industry. Eight real-life instances of the DSPSC from seven Northern European transport companies are acquired from TGE’s commercial system for this paper. It is estimated that the DSPSC accounts for 60% of a transport company’s operational expense. This paper proposes a column generation approach that attempts to minimize operational expense. Different instances present different challenges and being able to consider all conditions that arise in practice in order to improve efficiency of transport systems is seen as the primary contribution of this paper. This paper is organized as follows. Section B.2 gives a description of the existing literature related to the column generation technique and the DSP. In Section B.3, a detailed description of the various labour regulations is given and the DSPSC is described with the help of a mathematical model. Section B.4 introduces the proposed column generation framework for solving the DSPSC. Section B.5 details the computational study based on experimental tests performed on instances from seven Northern European transport companies. Finally, Section B.6 concludes the paper and addresses future directions of research.

B.2 Related Literature

The column generation has commonly been used in the literature to solve large routing and scheduling problems that deal with huge number of vari-
ables (Lübbecke and Desrosiers, 2005). Some applications of column generation include DSP (Desrochers and Soumis, 1989), vehicle scheduling problem (Desaulniers et al., 1998 and Hadjar et al., 2006) and vehicle routing problem (Desrochers et al., 1992). In most applications, the master problem of the column generation method is a set partitioning or set covering problem (SCP) with side constraints (Irnich and Desaulniers, 2005), and the subproblem is responsible for generating variables for the master problem. Desrosiers et al. (1984) was one of the first attempts to embed column generation within a linear programming based branch-and-bound framework to extract integer solutions for the vehicle routing problem with time windows (VRPTW). This method is commonly known as the branch-and-price (B&P) method. Desrosiers et al. (1984) modeled the subproblem as a shortest path problem (SPP) with time windows. This problem involves finding the least cost route between a source and a sink in a network while respecting specified time windows at each visited vertex in the network. An extension of the SPP is the shortest path problem with resource constraints (SPPRC), where the objective is to find the shortest (or least cost) path that respects all resource consumption constraints. The SPPRC has been applied to the VRPTW that includes vehicle capacity constraints, and was first modeled as the subproblem of the column generation method by Desrochers et al. (1992). Another variation of the SPPRC is the elementary shortest path problem with resource constraints (ESPPRC). This involves finding the shortest path that does not contain cycles i.e., a path does not revisit a vertex in the network. ESPPRC is known to be a NP-hard problem (Dror, 1994) and Feillet et al., 2004 is an example where the VRPTW is tackled as an ESPPRC.

SPPRC and its variations are most commonly solved by a labeling algorithm that is based on dynamic programming. The principle idea behind a labeling algorithm is to associate each possible partial path with a label that indicates the consumption of resources and to eliminate labels with the aid of domination rules. Based on the way the labels are managed, labeling algorithms can be classified into two categories, namely label-correcting and label-setting algorithms (Pugliese and Guerriero, 2013). A label-correcting approach is an extension of the Ford-Bellman algorithm, where each vertex in the network is considered many times and a label-setting approach is an adaptation of the Dijkstra’s Algorithm that considers each vertex once. Resource extension functions (REFs) are associated with arcs of the underlying network and are responsible for updating the consumption of resources along a partial path. REFs are viewed as the main tool for handling complex side constraints of routing and scheduling problems (Irnich, 2007). Examples of resources are cost, time, load and distance, and most REFs are linear. However, Irnich (2007) analyzed other non-linear REFs that arise in practice such as load dependent cost, where the cost of travelling along an arc depends on the load transported over the arc. Smith et al. (2012) considered the SPPRC with replenishment arcs that reset the accumulated resource along the partial path to zero. The authors stated that such situations
arise in the airline industry, where the resource represents the working period and replenishment arcs represent the overnight rests for the crew. Alternatively, for the aircraft routing problem, the resource represents the flight time and replenishment arcs represent the maintenance activities. For a detailed survey of the different variations of SPPRC and methods used to solve them, see Irnich and Desaulniers (2005) and Pugliese and Guerriero (2013).

For the DSP, each timetabled bus trip is associated with a constraint in the master problem, while the subproblem was first modeled as a SPPRC by Desrochers and Soumis (1989). The underlying network is an acyclic network and all paths are, hence, elementary. The vertices in the network represent the timetabled bus trips and the arcs denote the compatibility of two bus trips with respect to time and space. Additionally, driver activities such as break periods are placed on the arcs and each arc has a cost associated with it. A source and a sink vertex are added to the network, and a feasible path from the source to the sink represents a feasible schedule of a driver during the day. Resource consumption occurs on every arc and Desrochers and Soumis (1989) imposed four resource constraints that were primarily related to time, such as total duration of work and total paid time of a driver during the day. To find integer solutions, Desrochers and Soumis (1989) devised a B&P method that used a constraint branching idea of Ryan and Foster (1981). The authors tested the proposed method on a real-life instance from a transport company operating in an American city that had 167 bus trips. The column generation based method was able to produce a saving of 0.9% over a method that solved the SCP formulation with only a subset of feasible variables. The integer solution was obtained in less than one hour of computation time. Similarly, for an instance from a company operating in a British city that had 235 bus trips, the column generation based method was able to produce a saving of 2.4%. The computation time to find an integer solution was about three hours.

However, solving large instances of the DSP by column generation approaches have been reported in the literature as being computationally expensive due to the need to solve SPPRC at every iteration (Wren et al., 2003, Yunes et al., 2005 and Ibarra-Rojas et al., 2015). Yunes et al. (2005) devised a constraint programming approach to solve the subproblem and a dynamic programming technique to solve the SPPRC that was suggested by Desrochers and Soumis (1989). The authors tested the two aforementioned methods on instances from a transport company in the city of Belo Horizonte, Brazil. The sizes of the instances varied from 10 to 210 bus trips. The column generation algorithm based on dynamic programming could not solve instances with more than 100 trips within a time limit of 24 hours. It was reported that 90% of the total computation time, on average, was spent on solving the SPPRC. However, the column generation based on constraint programming was able to solve the largest instance of 210 trips in less than 15 hours. Some authors, Steinzen (2007) and
Kliewer et al. (2012), have proposed acceleration techniques for solving the SP-PRC. The aforementioned authors initially considered a subset of resources as part of domination rules to discard labels, and gradually increased the domination rules. For large networks, Kliewer et al. (2012) considered an additional heuristic technique that limits the number of labels stored at each vertex of the network. Kliewer et al. (2012) reported that when the allowed number of labels was decreased by a factor 2, the computation time also decreased by factor 1.5 to 2.5 on average.

Several metaheuristic approaches have been proposed in the literature to solve the subproblem; some examples include genetic algorithms (Mauri and Lorena, 2007) and hyper-heuristics (Li et al., 2015). Li et al. (2015) generate all feasible variables for a given instance and several heuristics (local search, swap heuristic and greedy based heuristic) are devised to select a subset of feasible variables at each iteration of the column generation framework. For an instance with 500 trips, the number of feasible variables was found to be 8.3 million. The hyper-heuristic evaluates the different heuristics based on their selection of variables that contribute to the improvement of the objective at each iteration of the column generation framework.

Most of the works published in the DSP literature have focused on developing SCP based models as part of commercial software systems. Since the DSP is a practical problem, models have been developed in collaboration with planners and end-users of transport companies so that complexities that arise in practice are considered. Some examples include Smith and Wren (1988), Wren et al. (2003) and Portugal et al. (2009), where the authors heuristically generate a feasible subset of variables that comply with the labor regulations and the transport companies’ rules. The largest instance considered by Smith and Wren (1988) involved 309 bus trips and 4,892 variables in the SCP formulation. Portugal et al. (2009) tested the models on instances from several transport companies in Portugal that had up to 347 bus trips and 23,305 variables. Some authors have considered solving the SCP formulation by metaheuristic procedures. Lourenço et al. (2001) proposed tabu search and genetic algorithm, which were tested on instances with up to 348 bus trips and 74,000 duties in the SCP formulation. Similarly, Li and Kwan (2003) proposed a genetic algorithm that was tested on instances with up to 1,873 trips and 50,000 duties in the SCP formulation.

A matheuristic based on adaptive large neighborhood search (ALNS) was proposed by Perumal et al. (2019) to tackle the DSPSC. Matheuristics are heuristic algorithms that are designed by combining metaheuristics and mathematical programming techniques (Boschetti et al., 2009). In the DSPSC, it was found that an integrated approach, where the drivers and the cars are scheduled simultaneously, is necessary to find a feasible solution. Column generation approaches have been applied to problems that integrate two scheduling problems. Some
B.3 Problem Description

examples include Friberg and Haase (1999) for the integrated vehicle and driver scheduling problem and Cordeau et al. (2001) for the integrated aircraft routing and crew pairing problem. In this paper, a state-of-the-art MIP solver (IBM ILOG CPLEX) and the matheuristic proposed by Perumal et al. (2019) are used as benchmark methods to evaluate the performance of the column generation approach.

B.3 Problem Description

B.3.1 Labor regulations

A set of bus trips, $T$, is given and serves as the input for the DSPSC. Each trip $t \in T$ has a departure time $\eta_t$, arrival time $\theta_t$, departure node $k_t$ and an arrival node $l_t$. A node is defined as a bus stop, including bus depot, in the city network that is visited by a bus and where a relief opportunity is available for the drivers. A relief opportunity implies that a driver change can occur at the location indicated by the node where a handover of the bus from one driver to another driver is completed. Only at specific nodes is the driver allowed to start and end his/her duty and $W$ denotes the set of driver depots. A driver duty is defined as the work of a driver for a day, and the duties have to comply with various labor rules and regulations. The labor rules and regulations are commonly influenced by public authorities and by negotiations carried out with labor unions. Transport companies often have contracts with multiple labor unions and the number of drivers under each labor union vary. Furthermore, few drivers are employed on a part-time basis, where work is carried out for a few hours in a week. Most commonly, the rules vary with each labor union and transport companies, hence, face a challenging task of creating a driver schedule that complies with all the rules in the contractual agreement made with each labor union. The set of duty types is denoted as $H$ and each $h \in H$ is defined by various labor rules and regulations such as:

1) **Maximum duration**
   Duration of a duty is defined as the period of time between the start of a driver’s duty and the end of the driver’s duty. A driver duty under duty type $h$ can never exceed maximum duration ($\text{maxDuration}^h$) limit.

2) **Minimum duration**
   The duration of a driver duty is at least $\text{minDuration}^h$.

3) **Maximum effective time**
   Effective time of a duty is defined as the total working time of a driver.
For instance, the total unpaid time of breaks is excluded from the effective time. The effective time of a duty can never exceed $max_{Effective}^h$.

4) **Maximum time without break**
The maximum time without break rule ensures that drivers have sufficient breaks during their working period. Breaks could either be coffee or meal breaks and the time between these breaks cannot exceed $max_{TimeWithoutBreak}^h$.

5) **Maximum number of long breaks**
A meal break is considered as a long break, whereas a coffee break is considered as a short break. As the names suggest, the duration of a meal break is longer than that of a coffee break. A duration of a meal break can vary between 20 minutes to 2 hours and a coffee break is few minutes (usually less than 20). A driver cannot have more than $max_{Longbreaks}^h$ meal breaks during his/her duty.

6) **Minimum number of long breaks**
The total number of meal breaks in a duty is at least $min_{Longbreaks}^h$.

7) **Maximum total duration of long breaks**
The meal breaks can vary in length and total duration of all meal breaks cannot exceed $max_{LongbreakDuration}^h$.

8) **Minimum total duration of long breaks**
The total duration of all meal breaks in a duty is at least $min_{LongbreakDuration}^h$.

9) **Maximum portion length**
Portion length is defined as the time period between two meal breaks. The portion length cannot exceed $max_{PortionLength}^h$. This rule is similar to the maximum time without break rule.

10) **Minimum portion length**
The minimum time between two meal breaks in a duty is at least $min_{PortionLength}^h$.

11) **Maximum number of vehicle changes**
Due to the various break rules, a driver carries out only a few consecutive bus trips before being relieved for a break. A driver duty typically consists of trips from multiple buses with sufficient breaks between them. Hence, a driver could potentially make several bus or vehicle changes during his/her duty. To avoid having too many vehicle changes during a duty, maximum number of vehicle changes ($max_{VehicleChanges}^h$) is imposed. This rule is, however, seen as an operational rule rather than a labor union rule.
A few labor unions enforce other special rules such as the **minimum break percentage**, which sets a minimum ratio of total duration of breaks to total working time. The aforementioned rules apply for most duty types, but there are duty types such as **Split duty type** and **EU duty type** that require very specific rules. A **Split duty type** is usually created to have a few drivers to cover bus trips during the morning and evening peak hours and drivers are allowed to have a very long break between the two shifts that can vary between 2 to 7 hours. The **EU duty type** rules are imposed by the European Union where the break rules differ if the driver covers a bus trip that is more than 50 kilometers.

### B.3.2 Staff cars

Most of the labor regulations are concerned with the break patterns for the drivers. Drivers have the ability to take breaks during relief opportunities at bus stops, but in most cases, only certain stops allow for breaks. This is mainly due to the availability of canteen and restroom facilities. Hence, the limited break options enforce drivers to travel between bus stops to have their breaks. The schedule of the buses also greatly influences driver break patterns, where if a stop that allows for breaks is frequently visited by the buses then the need for drivers travelling could potentially be reduced. In some cases, the breaks are allowed at specific stops that may not be visited by any buses or is limited.

Moreover, a driver could start/end his/her duty at stops that may not necessarily be the stops of the first/last bus trip of the corresponding duty and, as a consequence, drivers travel within the city to start/end their respective duties. Therefore, travel activities are essential to the DSP for finding a feasible driver schedule that respects all break rules of various labour unions. In practice, a driver travels by foot between two stops if the distance is short or by bus as a passenger. Another scenario that is evident in the public transport industry is the use of staff cars; transport companies have a fleet of cars at their depots that could be utilized by the drivers to reach designated stops. The use of staff cars adds flexibility to the driver schedule, however it complicates the DSP further. The cars that are being utilized by the drivers have to be returned to their respective depots at the end of the day and the number of cars at the depots is limited.
B.3.3 Mathematical model

Let $D$ denote the set of driver duties and the cost or total paid time of a duty $d \in D$ is represented as $c_d$. Binary matrix $A$ is defined, where $a_{td}$ is 1 if duty $d \in D$ covers trip $t \in T$. Another binary matrix $B$ is defined, where $b_{hd}$ is 1 if duty $d \in D$ is of duty type $h \in H$. A duty can be of only one duty type. The maximum number of duties that can be part of the driver schedule is denoted as $q$. Parameters $m_h$ and $n_h$ indicate the maximum and minimum number of duties of duty type $h \in H$. All duties in $D$ are checked for activities that include car travels, which are grouped into set $C$. Each car travel $i \in C$ has a departure time $\eta_i$, arrival time $\theta_i$, departure node $k_i$ and an arrival node $l_i$. Let $B$ be the set of nodes that can be visited by the drivers using staff cars and let $R \subset B$ be the set of car depots. A departure car travel is defined as a car travel that departs from depot $r \in R$. The set of departure car travels is denoted as $\hat{C} \subset C$, where $k_i = r \in R$, $l_i = b \in B \setminus R$ and $i \in C$. Similarly, an arrival car travel is defined as a car travel that arrives at depot $r \in R$. The set of arrival car travels is denoted as $\hat{C} \subset C$, where $k_i = b \in B \setminus R$, $l_i = r \in R$ and $i \in C$. Binary matrix $G$ is defined, where $g_{id}$ is 1 if duty $d \in D$ has car travel $i \in C$ and 0 otherwise. A car match is defined as a combination of a departure car travel from a depot and an arrival car travel to the same depot to form one round trip. Binary matrix $F$ is defined, where $f_{ij}$ indicates whether two car travels can be matched as one round trip, i.e. $k_i = l_j, l_i = k_j$, $\theta_i \leq \eta_j$, $i \in \hat{C}$ and $j \in \hat{C}$. $F$ denotes the set of all car matches, i.e. $F = \{(i, j) | f_{ij} = 1\}$. Each depot $r \in R$ has an upper limit $u_r$ on the number of cars that can be utilized. $E$ is defined as the set of all departure times of a staff car from the depot, i.e. $\{\eta_i\}$ where $i \in \hat{C}$, and $P_{er}$ denotes the set of all possible car matches that are active at time $e$ and associated with car depot $r$,

$$P_{er} = \{(i, j) | (i, j) \in F \land \eta_i \leq e \land \theta_j > e \land k_i = r\} \quad \forall e \in E, r \in R \quad (B.1)$$

Three decision variables are defined in the mathematical model. Binary variable $x_d$ indicates if duty $d \in D$ is selected as part of the schedule or not. Binary variable $y_i$ indicates if car travel $i \in C$ is used or not and binary variable $z_{ij}$ indicates if car travel $i \in \hat{C}$ is matched with car travel $j \in \hat{C}$ to form one round trip.

The mathematical formulation of the DSPSC is as follows:

$$\text{Minimize} \sum_{d \in D} c_d \cdot x_d \quad (B.2)$$
subject to,

\[
\begin{align*}
(\pi_t) & \quad \sum_{d \in D} \tilde{a}_{td} \cdot x_d \geq 1 & \forall t \in T \quad (B.3) \\
(\sigma) & \quad \sum_{d \in D} x_d \leq q \\
(\alpha_h) & \quad \sum_{d \in D} \tilde{b}_{hd} \cdot x_d \leq m_h & \forall h \in H \quad (B.5) \\
(\beta_h) & \quad \sum_{d \in D} \tilde{b}_{hd} \cdot x_d \geq n_h & \forall h \in H \quad (B.6) \\
(\gamma_i) & \quad \sum_{d \in D} \tilde{g}_{id} \cdot x_d \leq y_i \cdot M & \forall i \in C \quad (B.7) \\
(\delta_i) & \quad \sum_{d \in D} \tilde{g}_{id} \cdot x_d \geq y_i & \forall i \in C \quad (B.8) \\
(\xi_i) & \quad \sum_{j \in \hat{C}} \tilde{f}_{ij} \cdot z_{ij} = y_i & \forall i \in \hat{C} \quad (B.9) \\
(\zeta_j) & \quad \sum_{i \in \hat{C}} \tilde{f}_{ij} \cdot z_{ij} = y_i & \forall j \in \hat{C} \quad (B.10) \\
(\Delta_{er}) & \quad \sum_{(i,j) \in P_{er}} z_{ij} \leq u_{r} & \forall e \in E, r \in R \quad (B.11)
\end{align*}
\]

The objective of the DSPSC, given by (B.2), is to minimize the total cost of driver duties. Constraints (B.3) ensure that every bus trips is covered by at least one duty. Constraint (B.4) ensures that the total number of duties in the schedule does not exceed the maximum limit of \(q\). Constraints (B.5) and (B.6) ensure that the maximum and minimum limits on the number of duties per duty type are satisfied. In some cases, maximum and minimum average effective time of duties per duty type are also included as side constraints in the model. Constraints (B.7) ensure that a car travel is selected if it is utilized by one or more duties and \(M\) is considered as the seating capacity of staff cars. Constraints (B.8) ensure that a car travel is not selected if none of the duties utilize it. Constraints (B.9) together with constraints (B.10) ensure that a selected departure car travel from the depot is matched with an arrival car travel to the depot to form one round car trip. Constraints (B.11) ensure that the number of staff cars being utilized at all times during the day does not exceed the maximum number of cars available at each depot. For a linear program, the
integrality constraints (B.12) - (B.14) are relaxed and the corresponding duals of the constraints (B.3) - (B.11) are presented to its left as $\pi, \sigma, \alpha, \beta, \gamma, \delta, \xi, \zeta$ and $\Delta$.

### B.4 Methodology

For most applications, the number of possible duties is astronomical and the formulation (B.2) - (B.12) cannot be handled explicitly with all feasible duties. Column generation is a commonly used technique for problems with large number of variables. The column generation approach decomposes the problem into a master problem and one or more subproblems. Column generation is usually used in the context of linear programming where the integrality constraints (B.12) - (B.14) are relaxed and the master problem contains a reduced set of columns (or variables). Based on the dual information attained by solving the restricted master problem (RMP), only variables that have the potential of improving the solution would be generated and added to the RMP. The subproblem(s) is responsible for generating columns for the RMP. The set of duty, car travel and car match variables in the RMP are denoted as $D', C'$ and $F'$ respectively.

### B.4.1 Piece of work generation

The master problem in the column generation method is considered as a duty selection module and the subproblem(s) is considered as a duty generation module. Hence, all the labor rules and regulations that influence the feasibility of duties are defined in the subproblem(s) of the column generation method. A subproblem is constructed for each duty type $h \in H$ since the rules are different for each duty type and the master problem yields dual information for each duty type. Reduced cost of a duty $d \notin D'$ of duty type $h \in H$ is calculated as follows:

$$
\bar{c}_d = c_d - \sum_{t \in T} \bar{a}_{td} \pi_t - \sigma - \alpha_h - \beta_h - \sum_{i \in C} \bar{g}_{id} \gamma_i - \sum_{i \in C} \bar{g}_{id} \delta_i \quad (B.15)
$$

One approach that has been used in the literature (Freling et al., 2003, Huisman et al., 2005 and Kliwer et al., 2012) to generate duties is by enumerating and combining pieces of work. A piece of work (PoW) is a feasible sequence of consecutive trips of a bus as depicted with an example in Figure B.1. A duty consists of a number of PoWs and is typically restricted by maximum number of PoWs, which is equivalent to maximum vehicle changes (one vehicle change is
equivalent to two PoWs). In practice, most duties consist of up to three PoWs (two vehicle changes).

**Figure B.1:** A piece of work is defined as a feasible sequence of consecutive trips of a bus.

Figure B.2 shows an example of a duty that is made up of two PoWs. Most duty types require drivers to sign-on (start) and sign-off (end) duties at the same node. Relevant travels are added between two PoWs, and to the start and end of first and second PoW respectively to form valid duties. Depending on the duty type and its corresponding rules, short or long breaks are added between trips as depicted in Figure B.2.

**Figure B.2:** A duty is considered as a combination two or more pieces of work with relevant break and travel activities between them.

A PoW can be viewed as a partial duty and it is possible to enumerate all PoWs. However, due to the significantly large number of possibilities, it is very unlikely that all combinations of PoW that form feasible duties can be computed. By setting two as the maximum number of PoWs in a duty, all possible first and second PoWs are generated. For a large instance with 1900 bus trips, the number of PoWs for a duty type $h$ can be up to 50,000, and the total number of duties can be in the range 50,000-1.2 million. At each iteration of the column generation method, feasible combinations of PoWs are examined to see if they
yield duties with negative reduced cost. The reduced cost of a duty is computed as the sum of reduced cost of the PoWs it consists of and the reduced cost of the travel and break activities between the PoWs. Since the PoWs and its combinations are validated against the duty rules and regulations beforehand, the PoW based network is superior in terms of computation speed in a column generation setting. However, the disadvantage of the PoW network is that it is large and hence consumes memory. This has also been pointed out by Kliewer et al. (2012).

B.4.2 Shortest path problem with resource constraints

Another method of modeling the subproblem is the trip based network formulation for the SPPRC, which was discussed in Section B.2. The underlying network in the SPPRC is a directed acyclic network $G = (V, A)$, where each vertex $v \in V$ represents a bus trip and an arc $(i, j) \in A$ is created between two trips if they are compatible. A pair of trips $(t_1, t_2)$ is said to be compatible if $t_2$ can immediately be covered by a driver after $t_1$ while respecting all time and space constraints. All travel and break activities are placed on the arcs of the network. To ensure that the rule of drivers starting and ending their duties at the same node is satisfied, artificial source ($o$) and sink ($s$) vertices are created where arcs $(o, t_1)$ and $(t_1, s)$ represent the sign-on and sign-off of a duty. Therefore, a network $G_{h,w}^{h,w}$ is created for each duty type $h \in H$ and for each driver depot $w \in W$. For duty types that allow the driver to sign-on and sign-off his/her duty at different nodes, the closest driver depot is included in the source and sink arcs. Figure B.3 illustrates the underlying network with the representation of the dual information on the arcs and vertices. Additionally, cost, breaks and travels are denoted on every vertex and arc in $G$. Duties are generated by solving the SPPRC, where each feasible path from the source to the sink represents a feasible duty and the shortest feasible path is the duty with the most negative reduced cost.

SPPRC is solved by a dynamic programming algorithm, more precisely, by a label setting algorithm. Such an algorithm constructs partial paths from the source $o$ and a partial path is represented in the form of a label $(l)$ that holds information regarding the consumption of all resources along the path. An example of a resource could be time, where each activity on a vertex or an arc in $G$ has a limited duration and a duty is restricted by maximum duration. The labeling algorithm can be divided into three parts, namely 1) Resource extension functions (REFs), 2) Resource windows and 3) Domination rules and their descriptions for the DSPSC are:
1) Resource extension functions (REFs)

REFs are responsible for accumulating consumption of all resources along a path \( p \). Given an arc \((i, j) \in A\), the accumulated reduced cost of \( p \) at vertex \( j \) is calculated as \( \bar{Z}_j^p = \bar{c}_{ij} + \bar{c}_j \) where \( \bar{Z}_j^p \) is the accumulated reduced cost of path \( p \) at vertex \( i \), \( \bar{c}_{ij} \) is the reduced cost of arc \((i, j)\) and \( \bar{c}_j \) is the reduced cost of vertex \( j \). Similarly, REF of other resources are as follows:

- **Duration:** \( \bar{W}_j^p = \bar{W}_i^p + \bar{w}_{ij} + \bar{w}_j \), where \( \bar{w}_{ij} \) is the duration of arc \((i, j)\) and \( \bar{w}_j \) is the duration of vertex \( j \).

- **Effective time:** \( \bar{E}_j^p = \bar{E}_i^p + \bar{e}_{ij} + \bar{e}_j \).

- **Number of long breaks:** \( \bar{U}_j^p = \bar{U}_i^p + \lambda_{ij} \). There can utmost be one long break on an arc and \( \lambda_{ij} \) is, hence, a binary parameter which is 1 if there is a long break on \((i, j)\) and 0 otherwise.

- **Total long break duration:** \( \bar{P}_j^p = \bar{P}_i^p + \lambda_{ij}(\bar{\tau}_{ij} - \rho_{ij}) \). \( \bar{\tau}_{ij} \) represents the end time of long break on arc \((i, j)\) and \( \rho_{ij} \) represents the start time.

- **Number of vehicle changes:** \( \bar{R}_j^p = \bar{R}_i^p + \bar{r}_{ij} \), where \( \bar{r}_{ij} \) is a binary parameter that indicates if there is a vehicle change i.e., vertex \( i \) and \( j \) are from two different vehicles.

- **Portion length:**
  \[
  \bar{M}_j^p = \begin{cases} 
  \theta_j - \bar{\tau}_{ij}, & \text{if } \lambda_{ij} = 1 \\
  \bar{M}_i^p + \bar{w}_{ij} + \bar{w}_j, & \text{otherwise}
  \end{cases}
  \]
  , \( \theta_j \) is the end time of vertex \( j \).

- **Time since break:**
\[
\tilde{N}_j^p = \begin{cases} 
\theta_j - \max(\tau_{ij}, \Omega_{ij}), & \text{if } \lambda_{ij} = 1 \text{ or } \mu_{ij} = 1 \\
\tilde{N}_j^p + \bar{w}_{ij} + \bar{w}_j, & \text{otherwise}
\end{cases}
\]

\(\mu_{ij}\) is a binary parameter that indicates if there is a short break on arc \((i, j)\) or not and \(\Omega_{ij}\) is the end time of short break.

Label of \(p\) at \(j\) is denoted as \(l_j^p = (\bar{Z}_j^p, \bar{W}_j^p, \bar{E}_j^p, \bar{P}_j^p, \bar{M}_j^p, \bar{N}_j^p, \bar{R}_j^p)\). All the resources at the source \(o\) are initialized to zero i.e., \(l_o = (0, 0, 0, 0, 0, 0, 0)\). Multiple labels can be generated at each vertex with the aid of REFs and \(L_i\) denotes the set of all labels at vertex \(i\) \(\in V\), \(L_i = \{l_1^i, l_2^i, ..., l_n^i\}\).

2) **Resource windows**

To ensure feasibility of all paths generated in the labeling algorithm, a resource window \([l_l, u_l]\) is placed for all resources on all vertices of the network \(G^{h,w}\) \(h \in H, w \in W\), where \(l_l\) indicates the lower limit of a resource and \(u_l\) indicates the upper limit. Resource windows for the DSPSC are:

- Duration: \([0, \text{maxDuration}^h]\) \(\forall i \in V\setminus\{s\}\)
- Effective time: \([0, \text{maxEffective}^h]\) \(\forall i \in V\setminus\{s\}\)
- Number of long breaks: \([0, \text{maxLongbreaks}^h]\) \(\forall i \in V\setminus\{s\}\)
- Total long break duration: \([0, \text{maxLongbreakDuration}^h]\) \(\forall i \in V\setminus\{s\}\)
- Number of vehicle changes: \([0, \text{maxVehicleChanges}]\) \(\forall i \in V\setminus\{s\}\)
- Time since last break: \([0, \text{maxTimeWithoutBreak}^h]\) \(\forall i \in V\setminus\{s\}\)

Additionally, conditions are placed on arcs \((i, j)\) \(\in A\) such that if \(\lambda_{ij} = 1\) or \(\mu_{ij} = 1\) then,

\[
\tilde{N}_i^p + (\min(\rho_{ij}, \psi_{ij}) - \theta_i) \leq \text{maxTimeWithoutBreak}^h
\]

where \(\rho_{ij}\) is the start time of long break on arc \((i, j)\) and \(\psi_{ij}\) is the start time of short break.

- Portion length: \([0, \text{maxPortionLength}^h]\) \(\forall i \in V\setminus\{s\}\)

Feasibility checks are placed on the arcs \((i, j)\) \(\in A\) such that if \(\lambda_{ij} = 1\) then,

\[
\text{minPortionLength}^h \leq \tilde{M}_i^p + (\rho_{ij} - \theta_i) \leq \text{maxPortionLength}^h
\]

The above resource windows are also applied at the sink vertex \(s\). However, additional conditions are applied to satisfy lower limits of duration, long breaks, long break duration and portion length such as \([\text{minDuration}^h, \text{maxDuration}^h], [\text{minLongbreaks}^h, \text{maxLongbreaks}^h], [\text{minLongbreakDuration}^h, \text{maxLongbreakDuration}^h]\) and \([\text{minPortionLength}^h, \text{maxPortionLength}^h]\).
3) **Domination rules**

Multiple labels are associated with each vertex $i \in V$ and to avoid enumerating all feasible paths domination rules are applied to discard unpromising labels. Label $l_1^i$ at vertex $i$ dominates label $l_2^i$ if the following rules are satisfied:

\[
\begin{align*}
Z_i^1 & \leq Z_i^2 \\
W_i^1 & \leq W_i^2 \quad \text{(B.16)} \\
\bar{W}_i^1 & \geq \text{minDuration}^h \quad \text{and} \quad \bar{W}_i^2 \geq \text{minDuration}^h \quad \text{(B.18)} \\
\bar{E}_i^1 & \leq \bar{E}_i^2 \quad \text{(B.19)} \\
\bar{U}_i^1 & \leq \bar{U}_i^2 \quad \text{(B.20)} \\
\bar{U}_i^1 & \geq \text{minLongbreaks}^h \quad \text{and} \quad \bar{U}_i^2 \geq \text{minLongbreaks}^h \quad \text{(B.21)} \\
\bar{P}_i^1 & \leq \bar{P}_i^2 \quad \text{(B.22)} \\
\bar{P}_i^1 & \geq \text{minLongbreakDuration}^h \quad \text{and} \quad \bar{P}_i^2 \geq \text{minLongbreakDuration}^h \quad \text{(B.23)} \\
\bar{N}_i^1 & \leq \bar{N}_i^2 \quad \text{(B.24)} \\
\bar{M}_i^1 & \leq \bar{M}_i^2 \quad \text{(B.25)} \\
\bar{M}_i^1 & \geq \text{minPortionLength}^h \quad \text{and} \quad \bar{M}_i^2 \geq \text{minPortionLength}^h \quad \text{(B.26)} \\
\bar{R}_i^1 & \leq \bar{R}_i^2 \quad \text{(B.27)}
\end{align*}
\]

Algorithm 8 gives an overview of the labeling algorithm. All the vertices are initialized with an empty set of labels except $o$, where an initial label is created with zero consumption of all resources. $Q$ denotes a candidate list of vertices that are to be processed in the labeling algorithm and the list is initialized with $o$. A vertex $i$ is selected from $Q$ based on FIFO (First-In-First-Out) criteria and the labels at $i$, given by $L_i$, are considered one at a time (Line 4). Each label, $l_i$,
Algorithm 8: Labeling Algorithm

1. **Initialization:** \( L_i \leftarrow \emptyset \quad \forall i \in V \setminus \{o\}, \quad L_o \leftarrow \{l_o\}, \quad \hat{Q} \leftarrow \{o\}, \)
2. while \( \hat{Q} \neq \emptyset \) do
3. select \( i \in \hat{Q} \);
4. for \( l_i \in L_i \) do
5. for \( (i, j) \in A \) do
6. \( l_j \leftarrow \text{REFs}(l_i, (i, j)) \);
7. \( \text{checkFeasibility}(l_j) \);
8. if \( l_j \) feasible then
9. if \( j = s \) and \( l_j(\tilde{Z}_j) < 0 \) then
10. \( L_s \leftarrow L_s \cup l_j \);
11. else
12. \( \text{checkDomination}(l_j) \);
13. if \( l_j \) not dominated then
14. \( L_j \leftarrow L_j \cup l_j \);
15. \( \hat{Q} \leftarrow \hat{Q} \cup j \);
16. end
17. end
18. end
19. end
20. \( \hat{Q} \leftarrow \hat{Q} \setminus \{i\} \)
21. end
22. return \( L_s \)

is extended along each arc from \( i \), which is represented as \( (i, j) \), with the help of \( \text{REFs}() \). For each extension, a new label \( l_j \) is generated at \( j \) (Line 6). Function \( \text{checkFeasibility}() \) is used to discard infeasible labels (Line 7) by utilizing the resource windows at \( j \). If the feasible label has been extended to the sink vertex \( s \) and the reduced cost of the label is negative then the label is added to \( L_s \) (Line 9-10). However, if \( j \neq s \) then the function \( \text{checkDomination}() \) is applied to discard dominated labels at \( j \). If the new label \( l_j \) is not dominated then it is added to \( L_j \) and \( j \) is added to \( \hat{Q} \) (Line 13-15). The processed vertex \( i \) is removed from \( \hat{Q} \) (Line 21) and the algorithm terminates when \( \hat{Q} \) is empty. The labeling algorithm returns a set of labels at \( s \) with negative reduced cost as shown in Line 23.

By solving the SPPRC, feasible duties with negative reduced cost (\( \bar{e}_d < 0 \)) are generated and added to \( D' \) in the RMP. Hence, at each iteration of the column generation framework, dual information is updated in the SPPRC network and
suitable duties are generated. The PoW network described in Section B.4.1 is limited in its capability to explore the entire solution space of the DSPSC, where restrictions are imposed on the number of vehicle changes in order to avoid enumerating all possible duties. On the other hand, the SPPRC network enables the possibility of exploring solutions with more than one vehicle change and the labeling algorithm implicitly considers all possible duties. Solving the DSPSC by only using the PoW network leads to suboptimal solutions and the SPPRC network is required to achieve optimal solution. Additionally, the SPPRC network is advantageous in terms of memory consumption. However, as discussed in Section B.2, solving the SPPRC as part of the column generation framework is computationally expensive. Therefore, we propose to solve the DSPSC by utilizing the PoW network and the SPPRC network. Most often, few PoWs can be determined as feasible duties without any vehicle change, which is used to initialize the algorithm. The column generation framework could initially utilize the PoW network in order to quickly generate feasible duties to attain a lower bound (LB). Since the RMP is a linear programming (LP) model, the solution can be fractional and will therefore be a LB. When there are no more negative reduced cost duties in the PoW network, the column generation framework could then switch to the SPPRC network for improving the LP objective. For large problems, solving the SPPRC exactly, i.e. finding the shortest feasible path, can be a challenge. The domination rule in the labeling algorithm is responsible for discarding labels to avoid enumerating all feasible paths. However, if the domination rules are weak, i.e. labels cannot easily be dominated with respect to all resources, then not many labels are discarded early in the process. This causes more labels to be stored at each vertex of the SPPRC network, which affects the computational efficiency of the algorithm in terms of memory and speed. The literature for solving the DSP by tackling the SPPRC is limited and Desrochers and Soumis (1989), Kliewer et al. (2012) are some of the few authors that describe the SPPRC for solving the DSP. However, the authors had only four resources in their DSP. In the current DSPSC, there are seven resources (excluding reduced cost) and 12 domination rules given by equations (B.16) - (B.27). Therefore, solving to optimality might be impractical for large real life instances of the DSPSC. However, a heuristic labeling algorithm could be devised for improving the computational speed of the column generation algorithm and that provides near-optimal solutions. In the heuristic version of the labeling algorithm, we propose to relax some of the domination rules in order to increase the number of dominated labels. In particular, we relax minimum resource domination rules, equations (B.18), (B.21), (B.23) and (B.26). Another computational drawback of the column generation algorithm is the so-called tailing off effect (Lübbecke and Desrosiers, 2005), which is used to describe the last few iterations of the algorithm when the LP objective does not improve much. Generally, column generation terminates when there are no more negative reduced columns. However, for large problems, many authors such as Desaulniers et al. (1998) have used early termination criteria to avoid...
the tailing off effect. Since solving the SPPRC is known to be time consuming, the column generation algorithm is terminated when the LP objective does not improve by 0.001%.

B.4.3 Staff car subproblem

The PoW network and the SPPRC network are responsible for generating duties for the RMP. Let $D^I$ be the set of duties generated at each iteration of the column generation algorithm, i.e. $D^I = \{d|d \in D^\setminus D\}$. Additionally, the set of car travels generated can be known from $D^I$ and is denoted as $C^I = \{i|\bar{g}_{id} = 1, i \notin C', d \in D^I\}$.

At each iteration of the column generation algorithm, the staff car subproblem generates car matches for the RMP. The underlying network is a simple network, where the vertices represent car travels and the arcs represent possible matches between car travels. A network is created for each car depot $r \in R$. The reduced cost of a car match $\bar{c}_{(i,j)}$, where $(i,j) \notin F'$, is calculated as shown in equation (B.28). $\bar{E}$ is the set of times that the car match $(i,j)$ is active and is denoted as $\bar{E} = \{e|(i,j) \in P_{er}, e \in E, r \in R\}$. The set of car matches generated at each iteration is denoted as $F^I = \{(i,j)|\bar{c}_{(i,j)} < 0, (i,j) \notin F^I\}$. Car travels are also generated from the staff car subproblem and are represented in set $C^I = \{i|\bar{f}_{ij} = 1, i \notin C', (i,j) \in F^I\}$.

$$\bar{c}_{(i,j)} = -\xi_i - \zeta_j - \sum_{e \in \bar{E}} \Delta_{er} \quad \text{(B.28)}$$

Figure B.4 depicts the column generation framework for the DSPSC, where the RMP passes the dual information to the networks representing the driver and staff car subproblems, which then return duties, car travels and car matches to the RMP. The computational complexity of the RMP is expected to increase during the progress of the algorithm. The number of constraints in the RMP is estimated to be $|T| + (2 * |H|) + (3 * |C'|) + (|E| * |R|)$ and the number of variables in the RMP is $|D'| + |C'| + |F'|$. After each iteration, the number of variables and the number of constraints (corresponding to $C'$) will increase in the RMP.

In summary, the column generation algorithm devised for solving large real life instances of the DSPSC considers the following strategies:

- The different subproblems of the proposed approach include: a PoW network and a SPPRC network for the driver scheduling problem, and a
simple network for the staff car problem. The PoW network is initially used and the SPPRC network is utilized when there are no more negative reduced cost duties in the PoW network.

- To solve the SPPRC a heuristic labeling algorithm is proposed, where the minimum resource domination rules are relaxed.
- Early termination criteria is used to avoid *tailing off effect*.

### B.4.4 Integer solutions

In most cases column generation terminates with an LP solution that is not integral and the variables in the RMP take up fractional values. To attain integer solutions, B&P method is commonly used. At each node of the B&B tree, columns are generated until the optimal LP solution for the RMP is found. If the found LP solution satisfies integrality constraints then the upper bound (UB) is updated and the node is pruned, otherwise new branches are created. The B&P procedure is terminated when all nodes in the B&B tree have been processed. For more information on B&P procedure, the reader is referred to Barnhart et al. (1998).

A common branching strategy that has been applied for scheduling problems
is the strategy suggested by Ryan and Foster (1981). $\mathcal{D}(t_1, t_2) \subset D'$ is defined as the set of duties that covers trips $t_1$ and $t_2$. A pair of trips $(t_1, t_2)$ is chosen such that $0 < \sum_{d \in \mathcal{D}(t_1, t_2)} x_{\bar{d}} < 1$ for the LP solution. In an attempt to divide the solution space evenly, the following branches are created on the pair of trips $(t_1, t_2)$ that has fractional value closest to 0.5:

1-branch: $\sum_{d \in \mathcal{D}(t_1, t_2)} x_{\bar{d}} \geq 1$ \hspace{1cm} (B.29)

0-branch: $\sum_{d \in \mathcal{D}(t_1, t_2)} x_{\bar{d}} \leq 0$ \hspace{1cm} (B.30)

In the DSPSC, the pair of trips $(t_1, t_2)$ is considered to be from the same vehicle where $t_2$ is immediately covered after $t_1$. Hence, the 1-branch restricts duties from changing vehicle after completing $t_1$ and the 0-branch enforces duties to change vehicle after $t_1$ or not be covered at all. For the 1-branch, duties that do not execute $t_2$ after $t_1$ are removed from the RMP and the duties are forced to perform $t_1$ and $t_2$ together in the subproblem(s) network. For the 0-branch, duties that execute $t_2$ after $t_1$ are removed from the RMP and the arc between $t_1$ and $t_2$ in the subproblem(s) network is removed. However, the branching based on Ryan and Foster (1981) alone does not ensure integrality of the car travel variables in the RMP. The duty and car travel linking constraints, (B.7) and (B.8), in the mathematical model allow the car travel variables to take up fractional values in the solution. Hence, branching on the car travel variables is further required to attain an integer solution. In car travel branching, the variable $i \in C'$ that has a fractional value closest to 0.5 is selected and the following two branches are created:

1-branch: $y_i \geq 1$ \hspace{1cm} (B.31)

0-branch: $y_i \leq 0$ \hspace{1cm} (B.32)

Therefore, for the DSPSC, a strategy is proposed that performs Ryan&Foster branching and invokes car travel branching when there are no more pairs of trips to branch on. An alternative strategy would be to perform car travel branching first and follow this by Ryan&Foster branching.

The goal of the aforementioned branching strategies is to prove optimality by exploring the B&B tree. However, enumerating the entire B&B tree might be intractable for large problems and finding an integer solution in reasonable computation time can be challenging. For large problems, the focus is more on attaining a good quality integer solution quickly than proving optimality. One simple strategy is to solve the RMP as a MIP using a commercial solver with the variables generated at the root node of the B&B tree. Additionally, the following three heuristic branching strategies are considered:
1) Variable fixing

After solving a B&B node, the fractional variables in the RMP are processed and all the variables that have value above 0.8, i.e. \( x_d' \geq 0.8 \forall d' \in D' \) are fixed to 1. \( D' \) denotes the set of duties fixed in the RMP and \( T'_f = \{ t | \bar{a}_{td} = 1, t \in T, d \in D' \} \) is the set of trips in the fixed duties. Each \( t \in T'_f \) is removed from the subproblem(s) network. This is done in order to improve the speed of solving the subproblem(s) in the subsequent B&B node. The variable fixing procedure terminates either when an integer solution is found or when there are no more variables to fix. On circumstances when an integer solution could not be found, the integrality constraints (B.12)-(B.14) in the RMP are retained and the mathematical model is then solved using a commercial MIP solver. A similar strategy of fixing variables above a certain threshold was applied by Cordeau et al. (2001) for an integrated aircraft routing and crew scheduling problem.

2) Intertrips fixing

The intertrips fixing strategy is similar to 1-branch (B.29), except that multiple pairs of trips are considered to be fixed. Let \( A^c \) denote the set of all compatible pairs of trips that can be executed consecutively. For \((t_1, t_2) \in A^c\), \( t_1 \) and \( t_2 \) belong to the same vehicle and are known to be the predecessor and successor trips respectively. In an LP solution, multiple pairs of trips could have the same fractional value and all pairs of trips that have the closest value to 1 are fixed. The variables in the RMP and the subproblem(s) network are dealt in the same way as fixing a pair of trips in the 1-branch. However, this approach of fixing multiple pairs of trips reduces the size of the problem significantly. \( A^f \subset A^c \) denotes the set of pairs of trips that are fixed and \( T'_f \) denotes the set of trips that are fixed either as predecessors or successors. The intertrips fixing strategy terminates when all the fractional values of compatible pairs of trips are less than 0.9 or when the percentage of fixed trips exceeds 60, i.e. \( \frac{|T'_f|}{|T|} \times 100 > 60 \). Similar to the variable fixing strategy, a MIP solver is invoked when an integer solution could not be found.

3) Car match fixing

The fractional values of the car matches \((i, j) \in F'\) are processed and the car match variable \((z_{ij})\) that has the closest value to 1 is selected and fixed. The fixed car travels can be removed from the staff car suproblem. The strategy is terminated when all the fractional values of the car matches in the RMP are less than 0.5. The integrality constraints are retained and a MIP solver is used when an integer solution could not be found.

All the heuristic strategies explore the B&B tree in a depth-first manner without backtracking. Additionally, the column generation heuristic described in Section...
B.4.2 is utilized at each B&B node. Therefore, the heuristic approaches aim at quickly finding integer solutions for large real-life problems.

## B.5 Computational Study

### B.5.1 Instances

The aim of the computation study is to test the column generation algorithm and the different branching strategies for the DSPSC with instances of practical size. Table B.1 shows the sizes of real-life instances and since these instances pertain to several Northern European transport companies, they cannot be made publicly available. The instances in the large category are known to be the complete instances, which are used to extract small instances. The small instances are created to test the exact B&P approach.

$DK2$ is the largest instance with 1926 bus trips and five driver depots and, in practice, this instance does not involve the usage of staff cars. However, this instance was adapted to include car travels and the adapted instance is denoted by $DK2_{\text{staff cars}}$. Hence, a feasible solution in $DK2$ is also feasible in $DK2_{\text{staff cars}}$, but not vice versa. This adaptation was primarily done to compare the computational performance of the column generation algorithm for an instance with and without staff cars. $SE2$ is known to be the only instance with multiple staff car depots.

| Category | Instance      | $|T|$ | $|H|$ | $|W|$ | $|R|$ | $\sum_{r \in R} u_r$ |
|----------|---------------|------|------|------|------|------------------|
| Small    | $DK1_{\text{small1}}$ | 43   | 4    | 1    | 1    | 2                |
|          | $NO1_{\text{small}}$  | 73   | 1    | 1    | 1    | 1                |
|          | $NO2_{\text{small1}}$ | 74   | 3    | 1    | 1    | 1                |
|          | $SE1_{\text{small}}$  | 84   | 3    | 1    | 1    | 3                |
|          | $DK2_{\text{small1}}$ | 93   | 7    | 5    | 1    | 1                |
|          | $DK1_{\text{small2}}$ | 113  | 4    | 1    | 1    | 3                |
|          | $NO2_{\text{small2}}$ | 132  | 3    | 1    | 1    | 2                |
| Large    | $SE1$          | 204  | 3    | 1    | 1    | 6                |
|          | $FN1$          | 285  | 2    | 1    | 1    | 5                |
|          | $NO2$          | 487  | 3    | 1    | 1    | 3                |
|          | $DK1$          | 571  | 4    | 1    | 1    | 6                |
|          | $SE2$          | 623  | 4    | 3    | 3    | 15               |
|          | $NO1$          | 944  | 3    | 1    | 1    | 2                |
|          | $DK2_{\text{staff cars}}$ | 1,926 | 7    | 5    | 1    | 3                |
|          | $DK2$          | 1,926 | 7    | 5    | 0    | 0                |

**Table B.1:** Size of test instances. $|T|$ represents the number of trips, $|H|$ represents the number of duty types, $|W|$ represents the number of driver depots, $|R|$ represents the number of staff car depots and $\sum_{r \in R} u_r$ represents the total number of staff cars.
Table B.2 shows the number of variables generated for each instance. An Intel core i5-5287U @ 2.9 GHz machine with 16 GB memory is used to generate the variables. For some of the large instances, all variables cannot be enumerated as it leads to memory issues. Therefore, we decided to generate variables only from the PoW network, which is denoted as subset variables in Table B.2.

Table B.2: Number of variables from each instance. $|D|$, $|C|$ and $|F|$ represent the number of all feasible duties, car travels and car matches respectively. $|D'|$, $|C'|$ and $|F'|$ represent the number of duties, car travels and car matches generated from the PoW network. A '-' symbol indicates that the variables could not be generated within the limited memory.

### B.5.2 Results

The instances are tested with a state-of-the-art MIP solver (IBM ILOG CPLEX 12.8) and the matheuristic proposed by Perumal et al. (2019). The MIP solver is tested with all the variables generated and the subset of variables as shown in Table B.2. A maximum computation time of 18,000 seconds (five hours) is set for the MIP solver and the experiments are performed on an Intel Xeon E5-2680 v2 @ 2.80GHz with 128 GB memory. The matheuristic, which is based on ALNS, is tested only with the subset of variables and has a maximum number of iterations as the termination criteria. The matheuristic is run 10 times for each instance and the results presented are calculated as the average of 10 runs. The matheuristic experiments are performed on an Intel core i5-5287U @ 2.9 GHz machine with 16 GB memory. The overall CPU benchmarks reveal that the processor used for the MIP solver is approximately 3.4 times faster than
the processor used to test the matheuristic\textsuperscript{2}. The results of the MIP solver and the matheuristic are reported as observed in Table B.3. The best integer programming (IP) solution of the MIP solver or the average IP solution of the matheuristic is used as the benchmark for each instance. One should note that the matheuristic in Perumal et al. (2019) did not handle the side duty constraints (B.4) - (B.6). The matheuristic is tested without any adaptations and we, hence, believe that its performance may be affected by the side constraints.

<table>
<thead>
<tr>
<th>Category</th>
<th>Instance</th>
<th>MIP (All variables)</th>
<th>MIP (Subset variables)</th>
<th>Matheuristic (Subset variables)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>IP gap (%)</td>
<td>time (seconds)</td>
<td>IP gap (%)</td>
</tr>
<tr>
<td>Small</td>
<td>DK1_{small}</td>
<td>3,204</td>
<td>0.00</td>
<td>0.19</td>
</tr>
<tr>
<td></td>
<td>NO1_{small}</td>
<td>2,561</td>
<td>0.00</td>
<td>0.69</td>
</tr>
<tr>
<td></td>
<td>NO2_{small}</td>
<td>2,525</td>
<td>0.00</td>
<td>0.20</td>
</tr>
<tr>
<td></td>
<td>SE1_{small}</td>
<td>6,038</td>
<td>0.00</td>
<td>3.80</td>
</tr>
<tr>
<td></td>
<td>DK2_{small}</td>
<td>2,120</td>
<td>0.00</td>
<td>30.46</td>
</tr>
<tr>
<td></td>
<td>DK1_{small2}</td>
<td>8,643</td>
<td>0.00</td>
<td>6.22</td>
</tr>
<tr>
<td></td>
<td>NO2_{small2}</td>
<td>4,618</td>
<td>0.00</td>
<td>6.09</td>
</tr>
<tr>
<td>Large</td>
<td>SE1</td>
<td>14,664</td>
<td>0.00</td>
<td>212.56</td>
</tr>
<tr>
<td></td>
<td>FN1</td>
<td>19,902</td>
<td>0.00</td>
<td>891.58</td>
</tr>
<tr>
<td></td>
<td>NO2</td>
<td>38,994</td>
<td>0.35</td>
<td>18,004.90</td>
</tr>
<tr>
<td></td>
<td>DK1</td>
<td>36,882</td>
<td>3.18</td>
<td>18,001.60</td>
</tr>
<tr>
<td></td>
<td>DK2_{staff cars}</td>
<td>-</td>
<td>-</td>
<td>18,000.00</td>
</tr>
<tr>
<td></td>
<td>DK2</td>
<td>67,998</td>
<td>0.46</td>
<td>18,007.00</td>
</tr>
</tbody>
</table>

Table B.3: Results of the MIP solver and matheuristic. For the MIP solver, the best IP solution, optimality gap and computation time are reported. For the matheuristic, the best and worst IP solution found in the 10 runs are reported along with the average IP solution and average computation time. A '-' symbol indicates that an IP solution could not be found.

For small instances, optimal solutions are found by the MIP solver with all variables in quick computation time (\leq 30 seconds). The MIP solver with subset of variables provided solutions that are on average 2.02% from optimality. The matheuristic solutions are on average 3.06% from optimality, and the average computation time is around 93 seconds for the small instances.

The MIP solver with all variables proved optimality for two large instances (SE1 and FN1), and the optimality gap for SE2 instance is found to be 3.68\%. For these three instances, the MIP with subset of variables provided solutions that are on average 1.04\% from the solutions provided by the MIP with all variables. An IP solution could not be found for DK2_{staff cars} instance by the MIP solver within the time limit of five hours. The best IP solutions for instances NO2, DK1, NO1 and DK2 are provided by the MIP solver with subset of variables. The matheuristic could not tackle three instances (NO2, DK1 and NO1), where IP solutions could not be found consistently. However, the matheuristic provided

\textsuperscript{2}According to CPUbenchmark, the overall CPU rating of the processor used for the MIP solver is 15752 while that of the matheuristic is 4681.
IP solutions with an average computation time of one hour for \textit{DK2\_staff cars} instance, which could not be solved by the MIP solver. For the remaining large instances, the matheuristic solutions are on average 4.71\% from the best IP solutions found by the MIP solver. In conclusion, the MIP solver is superior to the matheuristic in terms of solution quality except for \textit{DK2\_staff cars} instance.

All the column generation based experiments are performed on the same processor as that of the matheuristic. Table B.4 shows the results of the exact B&P approach for small instances. Two branching strategies are tested; one where the Ryan\&Foster branching is followed by the car travel branching and the other where the car travel branching is followed by the Ryan\&Foster branching. A maximum computation time of 3,600 seconds (one hour) is set for the exact B&P algorithm. IP solutions could not be found for three out of the seven large instances within the time limit. For the instances that could be solved to optimality, it is observed that the number of B&B nodes processed by the strategy that performs car travel branching first is less than the strategy that performs Ryan\&Foster branching first. As a consequence, the strategy that performs car travel branching first is faster by factor 1.65 on average. However, the comparison between the results from the exact B&P and the results from the MIP solver (Table B.3) indicate that the computation times required by the exact approach to find IP solutions are immense and is, hence, considered impractical for solving large real-life instances.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Ryan&amp;Foster → car travel branching</th>
<th>Car travel → Ryan&amp;Foster branching</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B&amp;B nodes</td>
<td>IP</td>
</tr>
<tr>
<td>DK1_small1</td>
<td>1,897</td>
<td>\underline{3,204}</td>
</tr>
<tr>
<td>NO1_small1</td>
<td>2,251</td>
<td>\underline{2,561}</td>
</tr>
<tr>
<td>NO2_small1l</td>
<td>1,327</td>
<td>\underline{2,525}</td>
</tr>
<tr>
<td>SE1_small1</td>
<td>627</td>
<td>-</td>
</tr>
<tr>
<td>DK2_small1</td>
<td>35</td>
<td>\underline{2,120}</td>
</tr>
<tr>
<td>DK1_small2l</td>
<td>157</td>
<td>-</td>
</tr>
<tr>
<td>NO2_small2l</td>
<td>756</td>
<td>-</td>
</tr>
</tbody>
</table>

\textbf{Table B.4:} Results of Exact B&P for small instances. B&B nodes represents the number nodes processed in the B&B tree, IP represents the best integer programming solution, LP represents the best linear programming solution and gap represents the optimality gap. A '-' symbol indicates that an IP solution could not be found within the time limit.

Table B.5 shows the results of the different column generation based methods for large instances. One should note that the best LP solution in a heuristic column generation setting may not be found at the root node of the B&B tree and could be found in any of its succeeding nodes. The MIP solver is invoked for all instances and the maximum computation time of the solver is set to 10,800 seconds (three hours). The strategy that solves the root node of the B&B tree as a MIP found improved solutions for five instances and the average improvement is
found to be 0.88%. The aim of the branching decisions is to find an IP solution quickly; however, we found that if the decisions enforce many changes in the RMP and subproblems, then infeasibility maybe encountered. This observation is evident in the variable fixing strategy for instances DK1 and NO1. The variable fixing strategy found improved solutions for three instances. Although, IP solutions are found for all seven staff car instances by the car match fixing strategy, only two instances have improved solutions. The intertrips fixing strategy is found to be the best performing strategy where it provided improvements for six out of the eight instances. Interestingly, the two other instances (SE1 and FN1) are the smaller of the large instances for which the MIP solver could prove optimality as shown in Table B.3. The average improvement by the intertrips fixing strategy is found to be 1.45%. The computation time of the intertrips fixing strategy is comparable to that of the benchmark solutions and, in some cases, the intertrips fixing strategy is faster. However, the computation time of the intertrips fixing strategy for the DK2_staff cars instance is around 10 hours, which is much higher than that of the benchmark methods; the MIP solver has a computation time of five hours and the matheuristic has a computation time of one hour.

Summary statistics of the column generation based methods are shown in Table B.6. The total time (in seconds) spent by the methods on solving the RMP, driver scheduling subproblems and staff car subproblems are shown in Table B.6. DK2 is the only instance without any staff cars and it is observed that, on average, 78.8% of the total computation time of the column generation algorithm is spent on solving the driver scheduling subproblems and the remaining 21.2% is spent on solving the RMP. However, this behaviour is not evident for instances with staff cars and the complexity in solving the RMP tends to increase. For such instances, on average, 47.85% of the total computation of the column generation algorithm is spent on solving the RMP, 51.81% is spent on solving the driver scheduling subproblems and 0.34% is spent on solving the staff car subproblems. Overall, on average, 69.25% of the total time spent on solving the driver scheduling subproblems accounted for evaluating the domination rules of labels at all vertices of the SPPRC network.

B.5.3 Analysis of domination rules in the SPPRC

The computational study of the column generation algorithm shows that the domination rules in the SPPRC significantly contribute to the total time spent on solving the subproblems. To further analyze the computational complexity of the domination rules, the column generation algorithm is tested with increasing subsets of domination rules. DK2 is used as the test instance and, as shown in Table B.7, the number of domination rules is increased for each test. For
### Table B.5: Results of the four different column generation based methods for large instances

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Solution</th>
<th>Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SE1</td>
</tr>
<tr>
<td><strong>Root node</strong></td>
<td>IP</td>
<td>14,696</td>
</tr>
<tr>
<td></td>
<td>LP</td>
<td>14,677.63</td>
</tr>
<tr>
<td></td>
<td>gap (%)</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>time (seconds)</td>
<td>57.49</td>
</tr>
<tr>
<td></td>
<td>improvement (%)</td>
<td>-0.22</td>
</tr>
<tr>
<td><strong>Variable fixing</strong></td>
<td>IP</td>
<td>14,696</td>
</tr>
<tr>
<td></td>
<td>LP</td>
<td>14,677.63</td>
</tr>
<tr>
<td></td>
<td>gap (%)</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>time (seconds)</td>
<td>63.71</td>
</tr>
<tr>
<td></td>
<td>improvement (%)</td>
<td>-0.22</td>
</tr>
<tr>
<td><strong>Intertrips fixing</strong></td>
<td>IP</td>
<td>14,696</td>
</tr>
<tr>
<td></td>
<td>LP</td>
<td>14,677.63</td>
</tr>
<tr>
<td></td>
<td>gap (%)</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>time (seconds)</td>
<td>60.25</td>
</tr>
<tr>
<td></td>
<td>improvement (%)</td>
<td>-0.22</td>
</tr>
<tr>
<td><strong>Car match fixing</strong></td>
<td>IP</td>
<td>14,697</td>
</tr>
<tr>
<td></td>
<td>LP</td>
<td>14,677.63</td>
</tr>
<tr>
<td></td>
<td>gap (%)</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>time (seconds)</td>
<td>89.92</td>
</tr>
<tr>
<td></td>
<td>improvement (%)</td>
<td>-0.23</td>
</tr>
</tbody>
</table>

In these tests, the column generation algorithm is performed only at the root node of the B&B tree without the early termination criteria. One observation made from Table B.7 is that the time taken to evaluate the domination rules tends to increase with more domination rules in the SPPRC. However, domination rules such as maximum effective time (B.19), which is similar to maximum duration (B.17), seem to have no effect on the computation time and the LP objective. A label that dominates another label in terms of maximum duration (B.17) possibly also dominates in terms of maximum effective time (B.19). Another observation that is inferred from Table B.7 is that the LP objective is likely to improve with more and more domination rules. The improvement in LP objective by Test 8 when compared to Test 1 is found to be 0.68%; however, the total computation time taken to solve the subproblems is increased by factor 19. Test 9 in Table B.7 introduces the minimum duration (B.18) domination rule. It can be seen that the time taken to solve the subproblems increases tremendously and the column generation algorithm was terminated when it exceeded a time limit of 10,800 seconds (three hours). Therefore, for practical sized problems,
A Column Generation Approach for the DSPSC

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Instance</th>
<th>SE1</th>
<th>FN1</th>
<th>NO2</th>
<th>DK1</th>
<th>SE2</th>
<th>NO1</th>
<th>DK2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SE1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>FN1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>NO2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>DK1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>SE2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>NO1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>DK2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Root node

- **B&B nodes iterations**: 18, 25, 70, 49, 73, 53, 152, 114
- **|D|**: 8,436, 19,081, 30,798, 24,924, 17,063, 24,493, 52,959, 37,840
- **|C|**: 429, 635, 679, 1,347, 1,081, 507, 1,161, 0
- **|F|**: 5,980, 24,756, 13,479, 8,023, 14,336, 461, 10,301, 0
- **RMP (seconds)**: 7.77, 105.40, 155.80, 199.42, 190.75, 35.61, 5,813.24, 244.77
- **SP (seconds)**: 10.53, 169.99, 146.98, 405.64, 51.97, 199.91, 3138.59, 758.68
- **CP (seconds)**: 0.87, 2.72, 10.59, 1.92, 7.63, 0.10, 6.76, 0.00

### Variable fixing

- **B&B nodes iterations**: 2, 4, 3, 5, 10, 5, 5, 5
- **|D|**: 8,934, 23,508, 37,816, 29,612, 20,227, 30,625, 58,862, 39,796
- **|C|**: 435, 710, 688, 1,371, 1,090, 522, 1,182, 0
- **|F|**: 5,982, 24,756, 13,516, 8,079, 14,526, 487, 10,407, 0
- **RMP (seconds)**: 12.67, 230.38, 242.43, 280.48, 639.40, 136.92, 7,918.36, 420.79
- **SP (seconds)**: 9.11, 245.50, 248.06, 627.28, 68.54, 656.57, 6,301.01, 1395.71
- **CP (seconds)**: 0.79, 5.13, 7.91, 3.60, 18.49, 0.30, 13.96, 0.00

### Intertrips fixing

- **B&B nodes iterations**: 2, 27, 37, 60, 53, 55, 26
- **|D|**: 9,097, 23,007, 50,262, 33,086, 20,133, 35,063, 64,464, 40,474
- **|C|**: 437, 729, 714, 1,412, 1,095, 525, 1,186, 0
- **|F|**: 5,980, 24,756, 13,551, 8,189, 14,574, 487, 10,407, 0
- **RMP (seconds)**: 11.86, 500.27, 335.97, 996.89, 925.88, 283.80, 9,997.33, 475.60
- **SP (seconds)**: 11.59, 380.43, 559.55, 2,116.8, 187.18, 2560.08, 14,877.25, 2089.49
- **CP (seconds)**: 1.05, 13.26, 21.73, 12.14, 37.21, 0.30, 63.93, 0.00

### Car match fixing

- **B&B nodes iterations**: 5, 9, 2, 2, 9, 10, 2
- **|D|**: 9,804, 22,931, 37,107, 28,339, 19,650, 35,733, 56,818
- **|C|**: 440, 723, 684, 1,358, 1,095, 522, 1,175
- **|F|**: 5,980, 24,756, 13,516, 8,036, 14,886, 486, 10,327
- **RMP (seconds)**: 29.00, 455.45, 222.49, 254.12, 714.03, 407.07, 7,175.13
- **SP (seconds)**: 14.00, 288.29, 242.00, 632.49, 113.06, 3187.99, 4,098.47
- **CP (seconds)**: 1.41, 6.54, 8.20, 2.13, 14.90, 0.30, 9.61

Table B.6: Statistics. B&B nodes represents the number of nodes processed in the B&B tree, iterations represents the number of column generation iterations performed at the root node of the B&B tree, |D| represents the total number of duties generated, |C| represents the total number of car travels generated and |F| represents the total number of car matches generated. RMP represents the total time (in seconds) taken to solve the restricted master problem, SP represents the total time taken to solve the driver scheduling subproblems and CP represents the total time taken to solve the staff car subproblems.

As discussed in the Section B.2, some authors (Steinzen, 2007 and Kliewer et al., 2012) have considered gradually increasing the domination rules during the progress of the column generation algorithm to accelerate the SPPRC. A similar experiment is attempted and Figure B.5 illustrates the progress of the...
B.5 Computational Study

Test | Domination rules | Iterations | \(|D'|\) | LP | RMP (seconds) | SP (seconds) | Time taken to evaluate
domination rules (seconds)
--- | --- | --- | --- | --- | --- | --- | ---
1 | Reduced cost (B.16) | 412 | 33,219 | 67,099.75 | 313.93 | 148.21 | 4.54
2 | (+) Maximum duration (B.17) | 308 | 34,415 | 66,965.96 | 258.51 | 220.23 | 34.01
3 | (+) Maximum effective time (B.19) | 308 | 34,415 | 66,965.96 | 256.70 | 240.50 | 35.84
4 | (+) Maximum number of long breaks (B.20) | 242 | 35,553 | 66,966.25 | 290.39 | 415.91 | 156.58
5 | (+) Maximum total long break duration (B.22) | 242 | 35,553 | 66,966.25 | 288.63 | 371.10 | 140.59
6 | (+) Maximum time without break (B.24) | 215 | 36,898 | 66,860.26 | 270.49 | 1,090.41 | 666.35
7 | (+) Maximum portion length (B.25) | 196 | 39,277 | 66,671.14 | 240.14 | 1,555.15 | 1,110.73
8 | (+) Maximum vehicle changes (B.27) | 174 | 38,441 | 66,645.08 | 217.40 | 2,796.20 | 2,206.94
9 | (+) Minimum duration (B.18)* | 64 | 26,784 | 67,543.12 | 59.88 | 11,659.79 | 11,196.69

| Test | Domination rules | Iterations | \(|D'|\) | LP | RMP (seconds) | SP (seconds) | Time taken to evaluate
domination rules (seconds)
--- | --- | --- | --- | --- | --- | --- | ---
1 | Reduced cost (B.16) | 412 | 33,219 | 67,099.75 | 313.93 | 148.21 | 4.54
2 | (+) Maximum duration (B.17) | 308 | 34,415 | 66,965.96 | 258.51 | 220.23 | 34.01
3 | (+) Maximum effective time (B.19) | 308 | 34,415 | 66,965.96 | 256.70 | 240.50 | 35.84
4 | (+) Maximum number of long breaks (B.20) | 242 | 35,553 | 66,966.25 | 290.39 | 415.91 | 156.58
5 | (+) Maximum total long break duration (B.22) | 242 | 35,553 | 66,966.25 | 288.63 | 371.10 | 140.59
6 | (+) Maximum time without break (B.24) | 215 | 36,898 | 66,860.26 | 270.49 | 1,090.41 | 666.35
7 | (+) Maximum portion length (B.25) | 196 | 39,277 | 66,671.14 | 240.14 | 1,555.15 | 1,110.73
8 | (+) Maximum vehicle changes (B.27) | 174 | 38,441 | 66,645.08 | 217.40 | 2,796.20 | 2,206.94
9 | (+) Minimum duration (B.18)* | 64 | 26,784 | 67,543.12 | 59.88 | 11,659.79 | 11,196.69

Table B.7: Computational analysis of domination rules for instance DK2. Iterations represent the number of iterations performed at the root node of the B&B tree, \(|D'|\) represent the number of duties generated, LP represents the linear programming solution, RMP represents the total time (in seconds) taken to solve the restricted master problem and SP represents the total time taken to solve the driver scheduling subproblems. A '*' symbol indicates that the column generation algorithm was terminated when it exceeded a time limit of 10,800 seconds (three hours).

column generation algorithm for DK2 instance. The first phase of the algorithm is the PoW network and the domination rules are gradually introduced in the succeeding phases when there are no more negative reduced cost columns. An LB is quickly attained with the utilization of the PoW network and the LP objective is improved when the domination rules are introduced at each phase. It can be seen that the computational effort of the algorithm increases as the number of the domination rules increases. Additionally, the improvement found in the final phase of the algorithm is marginal and terminating the algorithm at earlier phases could be considered.

B.5.4 Sensitivity analysis

One of the drawbacks of the current DSPSC is that it does not consider the cost of staff car usage, which could include fuel cost or a fixed cost. The instances used in this paper have a fixed number of cars at the depots, which were given by the transport companies. However, one could question the total number of cars required in the final schedule. We perform a sensitivity analysis to study the impact of total number of cars on the solution quality and the computation time. SE1 is used as the test instance for the sensitivity analysis and Table B.8 shows the results when the instance is tested with different total number of cars. The problem is infeasible when there are no staff cars available, which signify that staff cars are essential for some instances in finding feasible solutions. The computation time tends to increase with increasing limitations on the total number of cars; the test with one staff car could not be solved to optimality within the time limit of five hours, whereas the tests with at least four staff
cars were solved in less than six minutes. Additionally, the IP solution could be improved when more staff cars are available at the depot. When comparing the results from the test with three staff cars and the test with four staff cars, the improvement is found to be 0.2%. However, this is not evident in all cases and increasing the total number of cars more than four does not affect the IP solution for SE1.

B.6 Discussion and Conclusion

In this paper, a column generation algorithm was proposed to tackle the DSPSC and several heuristic branching strategies were presented to find IP solutions for large real-life instances. A detailed computational study was performed with eight instances from seven Northern European transport companies. A commercial MIP solver and a matheuristic based on ALNS that was proposed by Perumal et al. (2019) were used as benchmark methods to evaluate the performance of the column generation algorithm. The column generation algorithm with intertrips fixing strategy as the heuristic branching strategy achieved improved solutions for six instances and the average improvement was found to
be 1.45%. The study further indicated that evaluating the domination rules in the SPPRC significantly contributed to the total computation time of the column generation algorithm. An acceleration heuristic technique that gradually introduces the domination rules during the progress of the algorithm was briefly tested and could be considered for instances in the future. Improving computation times of algorithms is also an important parameter from a practitioner’s perspective such that different scenarios could be analyzed quickly before selecting the final schedule to be implemented. This paper considered all challenges that arise in practice, which were faced by transport companies that operate in six cities. Hence, the work carried out in this paper could further be extended to test instances of cities with larger networks and more practical complexities.

As discussed in Section B.3, most of the labor regulations are concerned with the breaks for the drivers, and staff cars are required for finding feasible duties that adhere to the different break regulations. In the current DSPSC, the schedule of the buses are fixed and factors such as frequency of visits to break stops and the duration between consecutive bus trips may also influence the feasibility of driver duties. Hence, adapting the bus schedule to improve efficiency of the driver schedule could be considered. The vehicle scheduling problem (VSP) is concerned with finding a bus schedule with minimal bus operational expense and some of the work published in the literature that tackle this problem are Desaulniers et al. (1998), Hadjar et al. (2006) and Pepin et al. (2009). Work has also been carried out in integrating the vehicle and driver scheduling problem (IVDSP) in an attempt to further reduce the total operational expense. Some examples include Freling et al. (2003), Huisman et al. (2005) and Borndörfer et al. (2008). Incorporating the various labor regulations and the staff cars as part of the IVDSP is seen as the future area of research. This study also showed that the computational performance of the column generation algorithm dete-

<table>
<thead>
<tr>
<th>$\sum_{r \in R} u_r$</th>
<th>IP</th>
<th>gap (%)</th>
<th>time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>infeasible</td>
<td>-</td>
<td>2.93</td>
</tr>
<tr>
<td>1</td>
<td>17,577</td>
<td>15.37</td>
<td>18,000.60</td>
</tr>
<tr>
<td>2</td>
<td>15,081</td>
<td>2.48</td>
<td>18,001.12</td>
</tr>
<tr>
<td>3</td>
<td>14,693</td>
<td>0.00</td>
<td>16,849.23</td>
</tr>
<tr>
<td>4</td>
<td>14,664</td>
<td>0.00</td>
<td>354.81</td>
</tr>
<tr>
<td>5</td>
<td>14,664</td>
<td>0.00</td>
<td>263.19</td>
</tr>
<tr>
<td>6</td>
<td>14,664</td>
<td>0.00</td>
<td>226.56</td>
</tr>
<tr>
<td>Not assigned</td>
<td>14,664</td>
<td>0.00</td>
<td>187.05</td>
</tr>
</tbody>
</table>

Table B.8: Sensitvity analysis for instance SE1. $\sum_{r \in R} u_r$ represents the total number of staff cars, IP represents the integer programming solution and gap represents the optimality gap. ‘Not assigned’ indicates that the maximum staff car constraints (B.11) were not included.
riorates when the number of constraints increases during the progress of the algorithm. Evidence of this is indicated by the increase in computation time for solving the RMP for staff car instances. This has been pointed out earlier in the literature by Cordeau et al. (2001) and Mercier et al. (2005) for integrating aircraft routing and crew scheduling problems. The authors proposed a methodology that combines column generation and Benders decomposition to tackle inefficiencies of the column generation approach for large problems. Such methodologies should be explored in the future for integrating large scheduling problems.
Bibliography


and Crew Scheduling”. In: Transportation Science 35.4, pp. 375–388. DOI: 10.1287/trsc.35.4.375.10432. URL: http://pubsonline.informs.org/doi/abs/10.1287/trsc.35.4.375.10432.


Appendix C

Solution Approaches for Vehicle and Crew Scheduling with Electric Buses

Shyam S.G. Perumal\textsuperscript{a,b}, Twan Dollevoet\textsuperscript{c}, Dennis Huisman\textsuperscript{c}, Richard M. Lusby\textsuperscript{a}, Jesper Larsen\textsuperscript{a}, Morten Riis\textsuperscript{b}

\textsuperscript{a}Department of Technology, Management and Economics, Technical University of Denmark, Kgs. Lyngby, Denmark
\textsuperscript{b}QAMPO ApS, Aarhus, Denmark
\textsuperscript{c}Econometric Institute and Erasmus Center for Optimization in Public Transport (ECOPT), Erasmus University Rotterdam, The Netherlands

\textsuperscript{1}Abstract: The use of electric buses is expected to rise due to its environmental benefits. However, electric vehicles are less flexible than conventional diesel buses due to their limited driving range and longer recharging times. Therefore, scheduling electric vehicles adds further operational difficulties. Additionally, various labor regulations challenge public transport companies to find a cost-efficient crew schedule. Vehicle and crew scheduling problems essentially define the cost

\textsuperscript{1}To be submitted in Computers and Operations Research, December 2019
of operations. In practice, these two problems are often solved sequentially. In this paper, we introduce the integrated electric vehicle and crew scheduling problem (E-VCSP). Given a set of timetabled trips and recharging stations, the E-VCSP is concerned with finding vehicle and crew schedules that cover the timetabled trips and satisfy operational constraints, such as limited driving range of electric vehicles and labor regulations for the crew while minimizing total operational cost. An adaptive large neighborhood search that utilizes branch-and-price heuristics is proposed to tackle the E-VCSP. The proposed method is tested on real-life instances from public transport companies in Denmark and Sweden that contain up to 1,109 timetabled trips. The heuristic approach provides evidence of improving efficiency of transport systems when the electric vehicle and crew scheduling aspects are considered simultaneously. By comparing to the traditional sequential approach, the heuristic finds improvements in the range of 1.17-4.37% on average. A sensitivity analysis of the electric bus technology is carried out to indicate its implications for the crew schedule and the total operational cost. The analysis shows that the operational cost decreases with increasing driving range (120 to 250 kilometers) of electric vehicles.

**Keywords:** Public Transportation, Integrated Planning, Column Generation, Adaptive Large Neighborhood Search

### C.1 Introduction

The UN Paris climate agreement 2015 (United Nations Climate Change, 2015) that deals with mitigating greenhouse gas emissions worldwide influences policy makers and regulators to impose stringent emission standards. The European Union aims to reduce greenhouse gas emissions by at least 80% by 2050. Electric buses offer benefits such as improving overall air quality and reducing greenhouse gas emissions. The electric bus technology has been making its transition from niche to mainstream as its market share in Europe was estimated to be around 9% in 2018 (Transport and Environment, 2018). Most major cities in Europe are part of the C40 Fossil Fuel Free Street Declaration (C40 Cities, 2017) and have pledged to procure only zero-emission buses from 2025; Paris aims to electrify all of its 4,500 buses by 2025, all Dutch provinces are committed to procuring only zero-emission buses from 2025 (Transport and Environment, 2018) and Copenhagen city buses will be electric by 2025 (Copenhagen Capacity, 2019). Although electric buses provide significant environmental benefits, they are less flexible than the conventional diesel buses due to their limited driving range and
longer recharging times (Transport and Environment, 2018). Public transport companies and authorities are now faced with the challenge of making strategic decisions, for example on investment in battery package, charging infrastructure and placement of charging points in the city network.

Providing bus services requires solving several planning problems such as line planning, timetabling, vehicle scheduling and crew scheduling. In practice, these problems are solved in a sequential manner since solving the problems in one integrated step is too complex. Given a public transportation network that describes the underlying streets and bus stops in a city, Schöbel (2012) defines a line as a path along which a bus service is offered. The frequency of a line says how often the bus service is offered along the line within a given time period (e.g. an hour). The line planning problem determines a set of lines and their respective frequencies based on passenger demand. Next, timetabling determines the departure and arrival times of trips at bus stops of all lines. Subsequently, in the vehicle scheduling problem (VSP), the timetabled trips are assigned to the available buses such that every trip is covered by a bus. From the time a bus leaves the depot until it returns to the depot, it typically covers a sequence of trips. The schedule of a single bus is referred to as a block. Similarly, the work of a crew member or driver for a day is called a duty. The crew scheduling problem (CSP) aims to find an optimal set of duties that covers all bus trips and satisfies numerous labor union regulations. The VSP and CSP are the primary drivers of operational cost, and the public transport companies aim to minimize the total operational cost.

The VSP with multiple depots (MDVSP) is known to be an \(NP\)-hard problem (Bertossi et al., 1987) and has been studied extensively in the Operations Research (OR) literature. Some examples include Carpaneto et al. (1989), Ribeiro and Soumis (1994), Hadjar et al. (2006) and Pepin et al. (2009). Since the use of electric buses is on the rise in most countries, studies have been carried out on the electric vehicle scheduling problem (E-VSP) that determines the schedules of buses under limited driving ranges and fixed charging locations (Li, 2013, Wen et al., 2016, Adler and Mirchandani, 2017 and Kooten Niekerk et al., 2017). Rogge et al. (2018) focus on strategic electric bus planning that minimizes the total cost of ownership (TCO) of electric vehicle fleets. The TCO consists of the initial investments in vehicles and charging infrastructure, as well as the operational cost within a defined time period. Here, the crew cost is estimated to be the time-related operational cost of a bus. However, the true impact of electric vehicles on the CSP has not been studied in the OR literature to the best of our knowledge. An integrated approach that simultaneously handles the conventional vehicle and crew scheduling aspects has provided benefits such as reduction in number of drivers required and total operational cost when compared to the traditional sequential approach (Freling et al., 2003, Huisman et al., 2005 and Borndörfer et al., 2008). Therefore, given the additional opera-
ditional challenges of electric vehicles, integration of the E-VSP and CSP is an interesting field of research that could potentially contribute to improving the efficiency of transport systems.

In this paper, we introduce the integrated electric vehicle and crew scheduling problem (E-VCSP). Given a set of timetabled trips and recharging stations, the E-VCSP is concerned with finding vehicle and crew schedules that cover the timetabled trips and satisfy operational constraints, such as limited driving range of electric vehicles and labour regulations for the crew while minimizing total operational cost. An adaptive large neighborhood search (ALNS) algorithm (Ropke and Pisinger, 2006) is proposed to solve the E-VCSP. ALNS is a meta-heuristic that gradually improves an initial solution by destroying and repairing the solution iteratively using multiple destroy and repair methods. ALNS has gained popularity in recent years and has been applied to many transportation and scheduling problems (see e.g. Pisinger and Ropke, 2007 and Wen et al., 2016). Column generation, more precisely branch-and-price (B&P), is effective for solving large routing and scheduling problems (Lübbecke and Desrosiers, 2005). For the MDVSP, Pepin et al. (2009) studied the impact of utilizing a B&P heuristic as the repair method of an ALNS algorithm. The authors reported that combining the two methods provided high-quality solutions in short computation times. Similarly, in this paper, the ALNS algorithm relies heavily on B&P heuristic methods for exploration of large neighborhoods. Real-life instances from public transport companies operating in cities in Denmark and Sweden are acquired to study the E-VCSP and evaluate the proposed ALNS algorithm.

In summary, the contributions of this paper are i) the introduction and examination of the E-VCSP with aid of real-life instances, ii) the development of an ALNS algorithm, which utilizes B&P heuristic methods, to solve the E-VCSP and indicate potential benefits of integrating the two scheduling problems when compared to the traditional sequential approach, and iii) a sensitivity analysis that provides managerial insights into the implications of the electric bus technology for the crew schedule and the total operational cost.

The remainder of this paper is organized as follows. In Section C.2, we give a detailed description of the existing literature on the E-VSP and the integrated vehicle and crew scheduling problem (VCSP). Section C.3 describes the electric vehicle and crew operational rules considered in this study. In Section C.4, the E-VCSP is described with the help of a mathematical model. In Section C.5, the methods for computing lower bounds and upper bounds in short computation times for the E-VCSP are discussed. The proposed ALNS heuristic is described in Section C.6. Section C.7 evaluates the ALNS heuristic based on experiments performed on instances from public transport companies in Denmark and Sweden. The section also discusses the practical impact of the limited driving range
of electric vehicles on the crew schedule and the total operational cost. Finally, Section C.8 concludes the paper and addresses future research directions.

C.2 Related Literature

The scheduling of electric vehicles in public transportation has been extensively studied in the recent literature. Li (2013) address the single-depot VSP for electric buses with battery swapping or fast charging at given battery stations. The author presents an arc formulation of the problem that includes maximum distance before recharging or battery renewal constraints. Any resource constrained VSP is known to be \( \mathcal{NP} \)-hard (Bodin et al., 1983). The arc model is solved using a commercial mixed integer programming (MIP) solver. By applying Dantzig-Wolfe decomposition to the arc formulation, the problem is reformulated as a set partitioning problem or a path-based model. The model is solved by means of column generation and a variable fixing strategy is used for solving large instances. The author tested the arc model and the column generation method on instances from a bus company in Bay Area, California that contain up to 947 trips. Two different values of maximum operational distance of electric buses (120 and 150 km) are tested and the battery service time is set to 10 minutes. The author assumes that there exists one battery service station located at the depot and that it can service up to two vehicles at a time. For the large instances, the linear programming (LP) relaxation of the arc model is not solved to optimality by the commercial MIP solver in 12 hours. The column generation based method provided solutions that have an average optimality gap of 7% and the average computation time is found to be 72 hours. Adler and Mirchandani (2017) present the alternative-fuel MDVSP, where a set of fueling stations and fuel capacity for the vehicles are considered. A B&P algorithm and a heuristic that is based on the concurrent scheduler algorithm (Bodin et al., 1978) are proposed to solve the problem. Instances from a bus company in Phoenix, Arizona are used to evaluate both the methods. The buses are assumed to have a range of 120 km before needing to be refueled and the refueling time is set to 10 minutes. However, the B&P algorithm is tested only on subsets of the original data, which contained 4,373 timetabled trips. The subsets of the data had up to 72 trips, eight refuelling stations and four depots. The B&P algorithm took between two and 12 hours of computation time to solve the small instances. The heuristic took less than a second, but the average optimality gap is found to be 11.80%. Wen et al. (2016) address the E-VSP with full or partial recharging at any of the given recharging stations. The driving range of the vehicle is set to 150 km. The recharging process of the battery is assumed to be linear and a complete charging from empty to full takes two hours. The authors propose an ALNS heuristic for solving the
E-VSP. The method is tested on instances with up to 500 trips, eight depots and 16 stations. The authors use the optimal solutions of the MDVSP as lower bounds to evaluate the ALNS heuristic. The heuristic provided solutions in less than 15 minutes and the average gap is found to be less than 7%. Kooten Niekerk et al. (2017) incorporate non-linear charging behaviour of the batteries and time-dependent prices of energy in the E-VSP. The authors present column generation algorithms that are based on LP and Lagrangian relaxations. The methods are tested on instances from a bus company operating in Leuven, Belgium that contain up to 543 trips, one depot and four charging locations. Rogge et al. (2018) present the electric vehicle scheduling fleet size and mix problem with optimization of charging infrastructure, where the objective is to minimize the total cost of ownership of electric vehicle fleets. Given a set of timetabled trips and vehicle types, the problem determines the vehicle schedule to serve all trips and investment decisions such as the number of vehicles to buy per vehicle type. The charging infrastructure is considered to be installed at the depot and hence, the problem also focuses on the number of chargers to buy per depot. The authors propose a group genetic algorithm in combination with a MIP formulation. The authors tested the method on instances from two cities (Aachen, Germany and Roskilde, Denmark) and the instances had up to 200 trips.

In recent years, there has been an increasing focus on integrating two or more public transport planning problems. Several approaches have been proposed to integrate timetabling and the VSP, where the overall goal is to improve passenger service and reduce operational cost of vehicles (see e.g. Ibarra-Rojas et al., 2014 and Fonseca et al., 2018). Schöbel (2017) designs an iterative sequential algorithm to integrate line planning, timetabling and the VSP. The need to integrate the VSP and CSP was first recognized in the 1980s (Ball et al., 1983), since the crew cost was known to dominate the vehicle cost. For transport systems in Northern Europe, the crew cost contributes to approximately 60% of the total operational cost (Perumal et al., 2019). The cost structure necessitates the need for an integrated planning approach rather than a sequential approach which may lead to an inefficient crew schedule.

Methods in the OR literature for tackling the integrated vehicle and crew scheduling problem (VCSP) fall into two categories, namely partial and complete integration. Inclusion of crew considerations in the VSP and inclusion of vehicle considerations in the CSP are determined as partial integration methods. For an overview on partial integration methods, see Freling et al. (2003). Friberg and Haase (1999) propose an exact algorithm for the VCSP, where both the vehicle and crew aspects are formulated as a set partitioning problem. A branch-and-price-and-cut algorithm is proposed, where column generation and cut generation are combined in a branch-and-bound procedure. The authors tested the methodology on instances that contain to 30 trips. However, only
few instances with 20 trips could be solved to optimality within a reasonable computation time of five hours. Only the LP relaxation could be solved for the instance with 30 trips. Haase et al. (2001) also propose an exact approach for solving the single depot case of the VCSP. The authors present a set partitioning model with side constraints that only involves crew variables. Inclusion of vehicle cost and the side constraints in the formulation ensure that an overall optimal solution is found after deriving a compatible vehicle schedule. The model is solved by a B&P algorithm. For solving large instances, a heuristic version is devised where the branch-and-bound tree is explored in a depth-first manner without backtracking. The method is tested on instances that contain up to 350 trips and the maximum integrality gap is found to be 1.5%. Freling et al. (2003) are the first authors to tackle complete integration of vehicle and crew scheduling problems of practical size. The authors are also the first to make a comparison between the integrated and the traditional sequential approaches. The mathematical formulation of the single depot VCSP is a combination of a quasi-assignment formulation for the VSP, and a set partitioning formulation for the CSP. The authors propose a solution approach that is based on Lagrangian relaxation in combination with column generation. The columns that are generated to compute the lower bound are used to construct a feasible solution either by heuristic approaches or using a commercial MIP solver. The authors used subgradient optimization to solve the Lagrangian dual problem approximately. Instances from RET, the public transport company in Rotterdam, the Netherlands, were obtained to test the proposed method. The instances contained up to 238 trips. The primary objective was to minimize the sum of vehicles and drivers used in the schedule. The proposed integrated approach provided savings of at most one driver when compared to the sequential approach. Huisman et al. (2005) consider the VCSP with multiple depots and extend the solution approach proposed by Freling et al. (2003). Real life instances from the largest bus company in the Netherlands were obtained to test the method. The instances had up to 653 trips and four depots. The results showed that the integrated approach has a significant impact when compared to the traditional sequential approach; for an instance with 220 trips, the integrated approach provided a solution with 10 drivers less than that of the sequential approach. Borndörfer et al. (2008) propose a similar method to that of Freling et al. (2003) and Huisman et al. (2005) to solve the VCSP. However, the authors use bundle techniques for the solution of Lagrangian relaxations. The authors applied the proposed method to real life instances of a German city, Regensburg, which had up to 1,414 trips. The objective function used by the authors is a mix of fixed and variable vehicle cost, fixed cost and paid time of duties and various penalties related to operational requirements of the CSP. For the largest instance, an improvement of 3.69% in the objective value was provided by the integrated approach when compared to that of the sequential approach.

Steinzen et al. (2010) present a new modeling approach for the VCSP that is
based on a time-space network representation of the underlying vehicle scheduling problem. The authors also propose a column generation method based on Lagrangian relaxation. Furthermore, a heuristic B&P method is proposed to construct feasible solutions. The authors tested the proposed solution approach on randomly generated instances considered by Huisman et al. (2005) that contain up to 400 trips and four depots. The proposed solution approach outperforms the approaches of Huisman et al. (2005) and Borndörfer et al. (2008) in terms of solution quality and solution time. Kliewer et al. (2012) investigate an extension of the VCSP that involves the application of time windows, where the timetabled trips can be shifted within a specified interval. The extension can be seen as a partial integration of timetabling into the VCSP that offers further flexibility for scheduling vehicles and crews. The authors extend the solution approach proposed by Steinzen et al. (2010) and state that trip shifting enables additional break possibilities between trips for the drivers. Even with very short time windows (up to four minutes) for the timetabled trips, the authors show that enormous savings in the number of planned vehicles and drivers can be achieved.

In summary, given the imminent challenges of electric buses, the E-VSP is a growing area of research. Additionally, potential efficiency improvements provided by integrating the VSP and CSP have motivated researchers to explore methods to solve the VCSP. We hope that, by tackling the E-VCSP, we contribute valuable findings to the OR community and the public transport industry.

C.3 Problem Description

Let $L$ be the set of lines and $T$ be the set of timetabled trips that need to be covered by vehicles and drivers. Each line $l \in L$ consists of a set of timetabled trips denoted by $T_l \subseteq T$. Each trip $t \in T$ is defined by a departure bus stop, arrival bus stop, departure time and arrival time. A block, which represents the schedule of a vehicle, covers a subset of trips. The VSP determines the set of blocks that covers all timetabled trips $T$. Each block often starts with an empty move, i.e. a move without passengers, from the depot and ends with an empty move to the depot. Additionally, empty moves are placed between trips that do not end and start at the same bus stop. These empty moves are often referred to as deadheads. The cost of a block includes a fixed cost and a variable cost that is based on the total distance, in kilometers (km), covered by the vehicle during the day. In a multiple depot setting, the VSP typically includes only one operational constraint that requires the vehicles to start and end at the same depot. In this study of the E-VSP, similar to Li (2013), only
one depot is investigated and the following operational rules are considered to ensure feasibility of an electric vehicle or E-vehicle schedule:

1. **Maximum distance without recharging**
   An E-vehicle can cover a limited distance (km) before it has to be recharged at any of the given recharging stations.

2. **Minimum recharging duration**
   Traditional plug-in charging at the depot and pantograph charging at bus stops are two most common charging infrastructures (Transport and Environment, 2018). In this paper, only the depot charging facility is considered. Furthermore, this paper considers only full battery recharging with a minimum recharging duration.

Next to the timetable trips, each deadhead needs to be assigned to a driver if it is in the vehicle schedule. The cost of a driver duty includes a fixed cost and a variable cost that is based on the number of hours the driver works during the day. Labor unions often impose various regulations that govern the working conditions of the drivers. The following operational rules are considered to ensure feasibility of a crew schedule:

1. **Maximum duration of a duty**
   Duration of a duty is defined as the period of time between the start and end of a driver’s duty. The duration of a driver’s duty can never exceed a certain limit. Additionally, drivers are required to start and end their duties at the same depot. A driver could travel by foot or car between bus stops and the depot in order to start/end duty. However, the travel activities are also considered to be part of the driver’s duty.

2. **Minimum break duration**
   A driver often has multiple break periods during the day. A minimum duration is considered for a break and, in most cases, breaks are allowed only at certain bus stops.

3. **Maximum duration without break**
   The maximum duration without break rule ensures that drivers have sufficient breaks during their working period.

4. **Maximum number of vehicle changes**
   A driver duty typically consists of trips on multiple vehicles. A driver could potentially make several vehicle changes during the day. Too many vehicle changes could lead to operational challenges and hence, a maximum number of vehicle changes per driver duty is imposed. Essentially, a vehicle
change interchanges responsibilities for a vehicle between two drivers. A *takeover* is described as an event when a driver accepts responsibility for the vehicle. A *handover* is described as an event when a driver is relieved of his/her responsibility for the vehicle.

5. **Continuous attendance of vehicles**

   An *idle time* is defined as the time a vehicle is idle at a bus stop other than the depot. In most cases, a vehicle is idle for a brief period between the end and start of two consecutive trips. The continuous attendance of vehicles rule ensures that a driver is always present when a vehicle is outside the depot. In this study, it is assumed that drivers are allowed to have a break while attending to a vehicle when it is idle and the minimum break duration rule is satisfied. Furthermore, since only depot charging is considered, a driver need not attend to the vehicle when it is being recharged.

The aim of the E-VCSP is to minimize the total cost of E-vehicle and crew schedules that cover the set of timetabled trips \( T \) and satisfy all of the operational rules.

### C.4 Mathematical Formulation

Two network models are created for the E-VCSP; one for the E-vehicles and one for the crew. The underlying network of the E-vehicles is a directed acyclic network \( G^{EVSP} = (V^{EVSP}, A^{EVSP}) \), where each vertex \( v \in V^{EVSP} \) represents a trip and an arc \((i, j) \in A^{EVSP}\) indicates that trip \( j \) can immediately be covered by a vehicle after performing trip \( i \). It is assumed that if the idle time is more than a certain limit, i.e. an hour, and there is enough time for the vehicle to be fully recharged, then the corresponding deadheads to and from the depot and recharging activities are performed between two trips. All deadheads, idle times and recharging activities are placed on the arcs of the network. Additionally, artificial source \( o^{EVSP} \in V^{EVSP} \) and sink \( s^{EVSP} \in V^{EVSP} \) vertices are created. An arc from \( o^{EVSP} \) denotes the first pull-out deadhead from the depot and an arc to \( s^{EVSP} \) denotes the last pull-in deadhead to the depot of a vehicle. A path that respects the recharging requirements from \( o^{EVSP} \) to \( s^{EVSP} \) represents a block. \( F \) denotes the set of all deadheads and \( I \) denotes the set of all idle times in the E-vehicle network.

The crew network is a directed acyclic network \( G^{CSP} = (V^{CSP}, A^{CSP}) \) where each vertex \( v \in V^{CSP} \) corresponds to a departure/arrival bus stop and departure/arrival time of a trip or deadhead. Each arc \((i, j) \in A^{CSP}\) represents a
movement of the driver in time or in space and time dimensions. Additionally, artificial source \( o^{CSP} \in V^{CSP} \) and sink \( s^{CSP} \in V^{CSP} \) vertices are created. An arc from \( o^{CSP} \) denotes a driver duty sign-on activity at the depot and an arc to \( s^{CSP} \) denotes a driver duty sign-off activity at the depot. Furthermore, travel and break activities for the drivers are placed on the arcs. A path that satisfies the duty requirements from \( o^{CSP} \) to \( s^{CSP} \) represents a duty. Some arcs in the network represent a driver driving or attending to a vehicle, whereas other arcs represent vehicle changes. Figures C.1, C.2 and C.3 depict examples of an idle time, deadhead, recharging activity and a vehicle change in the E-vehicle and crew scheduling network.

**E-Vehicle scheduling network**

![E-Vehicle scheduling network](image)

**Crew scheduling network**

![Crew scheduling network](image)

**Figure C.1:** An example of a deadhead and an idle time between two trips in the E-vehicle and crew scheduling network. The figure also shows an example of a takeover and handover event after a trip or deadhead in the crew scheduling network.

Let \( B \) be the set of all blocks and \( D \) be the set of all duties. The cost of a block \( b \in B \) is represented as \( c^1_b \) and \( c^2_d \) denotes the cost of a duty \( d \in D \). Binary matrix \( A^1 \) is defined, where \( a^1_{bt} \) is 1 if block \( b \in B \) covers trip \( t \in T \). Similarly, \( A^2 \) is a binary matrix, where \( a^2_{td} \) is 1 if duty \( d \in D \) covers trip \( t \in T \). \( A^3 \) is a binary matrix, where \( a^3_{bf} \) is 1 if block \( b \in B \) contains deadhead \( f \in F \). \( A^4 \) is a binary matrix, where \( a^4_{fd} \) is 1 if duty \( d \in D \) contains deadhead \( f \in F \). \( A^5 \) is a binary matrix, where \( a^5_{ib} \) is 1 if block \( b \in B \) contains idle time \( i \in I \). \( A^6 \) is a binary matrix, where \( a^6_{id} \) is equal to 1 if duty \( d \in D \) contains idle time \( i \in I \). Two types of decisions variables are defined in the mathematical model. Binary variables \( y_b \) indicate whether block \( b \in B \) is selected as part of the schedule or
An example of a recharging activity at the depot is shown in the E-vehicle and crew scheduling network. A driver need not attend to a vehicle while it is being recharged.

Figure C.2: An example of a recharging activity at the depot is shown in the E-vehicle and crew scheduling network. A driver need not attend to a vehicle while it is being recharged.

not. Binary variables $x_d$ indicate whether duty $d \in D$ is selected as part of the schedule or not. The mathematical formulation of the E-VCSP is as follows:

$$\text{Minimize } \sum_{b \in B} c_b^1 \cdot y_b + \sum_{d \in D} c_d^2 \cdot x_d$$  \hspace{1cm} (C.1)$$

subject to,

$$\sum_{b \in B} a_{tb}^1 \cdot y_b = 1 \hspace{1cm} \forall t \in T$$  \hspace{1cm} (C.2)$$

$$\sum_{d \in D} a_{td}^2 \cdot x_d = 1 \hspace{1cm} \forall t \in T$$  \hspace{1cm} (C.3)$$

$$\sum_{d \in D} a_{td}^4 \cdot x_d - \sum_{b \in B} a_{tb}^3 \cdot y_b = 0 \hspace{1cm} \forall f \in F$$  \hspace{1cm} (C.4)$$

$$\sum_{d \in D} a_{td}^6 \cdot x_d - \sum_{b \in B} a_{tb}^5 \cdot y_b = 0 \hspace{1cm} \forall i \in I$$  \hspace{1cm} (C.5)$$

$$y_b \in \{0, 1\} \hspace{1cm} \forall b \in B$$  \hspace{1cm} (C.6)$$

$$x_d \in \{0, 1\} \hspace{1cm} \forall d \in D$$  \hspace{1cm} (C.7)$$
The objective of the E-VCSP, given by (C.1), is to minimize the total operational cost. Constraints (C.2) and (C.3) ensure that every trip is covered by exactly one block and one duty respectively. Constraints (C.4) ensure that duties are selected to cover deadheads that are utilized by blocks in the solution. Constraints (C.5) satisfy the continuous attendance of vehicle rule, where a duty is selected to cover an idle time corresponding to a block in the solution. The model contains $|B| + |D|$ variables and $2|T| + |F| + |I|$ constraints. In practice, often additional side constraints such as maximum number of allowed blocks and duties are present.

C.5 Lower Bounds and Fast Upper Bounds

In this section, we discuss methods for computing lower bounds and fast upper bounds for the E-VCSP. An integrated approach that solves the LP relaxation of the integrated mathematical model, given by Equations (C.1) - (C.7), to optimality is described in Section C.5.1. The optimal LP objective value given by the integrated approach is denoted as $Z_{Integrated}$. Another method of computing a lower bound is an independent approach, where the optimal LP solutions of the E-VSP and the CSP are found independently and their respective optimal LP objective values are added afterwards to give an overall lower bound for the E-VCSP. The independent approach is described in Section C.5.2 and the resulting lower bound is denoted as $Z_{Independent}$. However, the integrated approach is considered to provide stronger or improved lower bounds when compared to
that of the independent approach. In this paper, we denote $Z_{LB}$ as the best known lower bound for a given instance of E-VCSP.

A method to compute an upper bound for the E-VCSP in short computation time is the traditional sequential approach that solves the E-VSP first and then the CSP. Section C.5.3 describes the sequential approach and the solution provided by the sequential approach is denoted as $Z_{Sequential}$. The potential benefit of integration is measured by comparing the solution of the traditional sequential approach ($Z_{Sequential}$) to the best known lower bound ($Z_{LB}$). The optimal objective value of the E-VCSP is denoted as $Z^*$. Figure C.4 gives an overview of the lower and upper bounds provided by the different methods, and $Z_{Independent} \leq Z_{Integrated} \leq Z^* \leq Z_{Sequential}$.

Figure C.4: Lower and upper bounds for the E-VCSP. $Z_{Independent}$ and $Z_{Integrated}$ denote the lower bounds provided by the independent and the integrated approaches respectively. $Z_{Sequential}$ represents the solution of the sequential approach. $Z^*$ denotes the optimal objective value, and $Z_{Independent} \leq Z_{Integrated} \leq Z^* \leq Z_{Sequential}$.

C.5.1 Integrated approach

The formulation (C.1) - (C.7) cannot be handled explicitly with all feasible blocks and duties. Column generation is commonly used to tackle problems with a large number of variables. The integrality constraints (C.6) and (C.7) are relaxed and the problem decomposes into a master problem and one or more subproblems. The master problem is initialized with a subset of variables (or columns) and is referred to as restricted master problem (RMP). On solving the RMP, the dual information is obtained: $\pi_t, \alpha_t, \sigma_f$ and $\gamma_i$ denote the duals of constraints (C.2) - (C.5), respectively. The subproblems are responsible for generating columns that are not included in the RMP, but have the potential of decreasing the RMP’s objective function value. The subproblems utilize the dual information from the RMP to identify negative reduced cost columns. Column generation is an iterative framework between the master and subproblem, which terminates when there are no more negative reduced cost columns. The set of block and duty variables in the RMP are denoted as $B'$ and $D'$, respectively. For the E-VCSP, there are two subproblems; one corresponds to $G^{EVSP}$ that generates block variables and the other corresponds to $G^{CSP}$ that generates
duty variables. The subproblems are extended into a shortest path problem with resource constraints (SPPRC) (see e.g. Irnich and Desaulniers, 2005) that is solved using a label-setting algorithm. The reduced cost of a block \( b \notin B' \) is calculated as follows:

\[
\tilde{c}_b^1 = c_b^1 - \sum_{t \in T} a_{tb}^1 \cdot \pi_t + \sum_{f \in F} a_{fb}^3 \cdot \sigma_f + \sum_{i \in I} a_{ib}^5 \cdot \gamma_i
\]  

(C.8)

Similarly, the reduced cost of a duty \( d \notin D' \) is calculated as follows:

\[
\tilde{c}_d^2 = c_d^2 - \sum_{t \in T} a_{td}^2 \cdot \alpha_t - \sum_{f \in F} a_{fd}^4 \cdot \sigma_f - \sum_{i \in I} a_{id}^6 \cdot \gamma_i
\]  

(C.9)

Since the RMP is an LP model, the block and duty variables can be assigned fractional values. Hence, in most cases, column generation terminates with an LP solution. However, the LP objective value \( Z_{\text{Integrated}} \) is determined to be a lower bound to the E-VCSP.

## C.5.2 Independent approach

An independent CSP (ICSP) can be formulated, where the vehicle considerations are completely ignored in the problem. The linking constraints, given by Equations (C.4) and (C.5), in the model are relaxed to decouple the E-VSP and ICSP. Since the vehicle schedule is not given, the possible set of duties is much larger in the ICSP formulation than in the CSP. A lower bound to the E-VCSP can be computed by independently solving the E-VSP and ICSP by column generation and adding their respective optimal LP objective values. The ICSP was proposed by Freling et al. (2003) to evaluate the solution of the sequential approach. The independent approach would provide lower bounds \( Z_{\text{Independent}} \) in short computation time. However, the bounds are not only believed to be weaker but are provably non-stronger than the lower bounds provided by the integrated approach described in Section C.5.1. Since the integrated approach deals with a large number of constraints, it might be intractable for solving large instances. In that case, optimal LP solutions cannot be found in reasonable computation time. On such cases, the independent approach could be used to obtain the best lower bound \( Z_{\text{LB}} \).

## C.5.3 Sequential approach

The sequential approach is commonly used in practice to compute a feasible solution \( Z_{\text{Sequential}} \). However, one should note that the sequential approach does
not always guarantee feasibility; i.e. a feasible crew schedule that satisfies all the labor regulations may not exist with respect to the vehicle schedule constructed in the first phase. The E-VSP model only includes the constraints (C.2) and the objective is to minimize the operational cost of vehicles, i.e. $\sum_{b \in B} c_b^1 \cdot y_b$. VSP and its extensions are also commonly solved by column generation approaches (see e.g. Ribeiro and Soumis, 1994). To attain integer solutions, the column generation method is embedded in a branch-and-bound (B&B) framework. This approach is known as the branch-and-price (B&P) method. For large problems, a heuristic version of the B&P method has been explored in the literature (Desaulniers et al., 1998, Pepin et al., 2009 and Li, 2013). In the heuristic version, the B&B search tree is explored in a depth-first manner without backtracking and variables ($y_b$) that have values above a certain threshold are fixed to 1 at each node of the tree. In this paper, all variables that have values greater than or equal to 0.8, i.e. $y_b \geq 0.8$, are fixed to 1. If there are no such variables, then the variable with the fractional value closest to 1 is selected and fixed.

All deadheads and idle times that are in the final solution of the E-VSP are passed to the CSP, where the objective is to minimize $\sum_{d \in D} c_d^2 \cdot x_d$. Hence, the CSP includes constraints (C.3), an adapted subset of constraints (C.4) to ensure that each deadhead in the E-VSP solution is covered by a duty and an adapted subset of constraints (C.5) to ensure that each idle time in the solution is covered by a duty. The CSP has also been commonly solved by column generation approaches (see e.g. Desrochers and Soumis, 1989). Similar to the E-VSP, a heuristic B&P version for solving the CSP is implemented to attain integer solutions. Duty variables ($x_d$) that have fractional values greater than or equal to 0.8 are fixed to 1. If there are no such variables, then the variable with the fractional value closest to 1 is selected and fixed.

Since the E-VSP and the CSP are computationally hard problems to solve, the sequential approach of solving the E-VCSP could still be very time consuming. The input for the E-VCSP is the set of trips $T$, which is partitioned into different lines that are given by $L$. Hence, a sequential approach could be applied for each individual line $l \in L$ that contains only a subset of trips $T_l$. Such an approach is seen as a construction heuristic that generates initial solutions in very short computation times.

C.6 Adaptive Large Neighborhood Search

In this section, we give a detailed description of our adaptive large neighborhood search (ALNS) heuristic for the E-VCSP. The solution obtained from the ALNS heuristic is denoted as $Z_{ALNS}$. In this study, the sequential solution
C.6 Adaptive Large Neighborhood Search

$(Z_{\text{Sequential}})$ is used as a benchmark to evaluate the performance of our ALNS heuristic.

ALNS is a local search framework that was proposed by Ropke and Pisinger (2006). The main idea of the ALNS heuristic is to move from one feasible solution to a neighboring solution by repeatedly selecting and applying a destroy and a repair method from a set of destroy and repair methods. The set of neighboring solutions of a current solution is referred to as a neighborhood. In ALNS, a neighborhood is implicitly defined by a destroy and a repair method. For more information on ALNS, see Pisinger and Ropke (2019).

Algorithm 9: Adaptive Large Neighborhood Search

1. **Initialization:**
2. $s \leftarrow \text{InitialSolution}(), s^* \leftarrow s$;
3. $\rho \leftarrow \text{InitializeWeights}()$;
4. $\Omega \leftarrow \text{InitializeScores}()$;
5. $\nu \leftarrow \text{InitializeAttempts}()$;
6. while stop criteria not met do
7.   Select neighborhood $n \in N$ using $\rho$;
8.   $s' \leftarrow \text{Repair}(\text{Destroy}(s, n))$;
9.   if $\text{Accept}(s, s')$ then
10.     $s \leftarrow s'$;
11.   end
12. if $f(s') < f(s^*)$ then
13.     $s^* \leftarrow s'$;
14. end
15. $\Omega \leftarrow \text{UpdateScores}(\psi, n)$;
16. $\nu \leftarrow \text{UpdateAttempts}(n)$;
17. if update criteria met then
18.     $\rho \leftarrow \text{UpdateWeights}(\Omega, \nu, \lambda)$;
19.     $\Omega \leftarrow \text{ResetScores}()$;
20.     $\nu \leftarrow \text{ResetAttempts}()$;
21. end
22. end
23. return $s^*$

Algorithm 9 gives an overview of the ALNS procedure. The current solution is denoted as $s$, the neighboring solution is denoted as $s'$ and the best solution is denoted as $s^*$. An initial solution is computed, which serves as input to the heuristic. The sequential approach for each individual line $l \in L$ that is described in Section C.5.3 can be used to obtain an initial solution quickly. Alternatively, the sequential approach can be applied to the entire problem with all trips $T$ to obtain an initial solution, which is known to take more time. A set of
neighborhoods $N$ is defined and each $n \in N$ is assigned a modifiable weight $\rho^n$. A neighborhood $n \in N$ is selected to perform a destroy and repair operation on the current solution at each iteration of the ALNS heuristic. The probability of a neighborhood being selected is determined as shown in Equation (C.10). A roulette wheel principle is used to select a neighborhood at each iteration.

At the start of the heuristic, the weights of the neighborhoods are initialized to 1. For each $n \in N$, $\Omega^n$ denotes the accumulated score and $\nu^n$ denotes the number of times it has been selected. At each iteration, the chosen neighborhood $n$ is awarded a score of $\psi$. The quality of the neighboring solution $s'$ obtained is used to evaluate the chosen neighborhood. In this paper, the hill climber acceptance criterion is used, which only accepts improving solutions. A score of $\psi_1$ is rewarded to the selected neighborhood if it finds a new best solution, else a score of $\psi_2$ is given ($\psi_1 > \psi_2 \geq 0$). The rewarded score is added to $\Omega^n$. Every time the heuristic performs a certain number of iterations ($\mu$), the weights of the neighborhoods are updated as follows,

$$\rho^n = (1 - \lambda) \cdot \rho^n + \lambda \cdot \frac{\Omega^n}{\nu^n} \quad \forall n \in N \quad (C.11)$$

After performing $\mu$ iterations, $\nu^n$ and $\Omega^n$ are reset to 0. The degree of change in weights is controlled by the reaction factor $\lambda \in [0, 1]$.

Typically, a maximum number of iterations is used as the stopping criterion of the heuristic. In this paper, the heuristic is terminated when the weights converge below a certain tolerance level, i.e. $\rho^n < \epsilon \quad \forall n \in N$. Additionally, the heuristic is terminated if it reaches a maximum computation time of $\text{max}_{\text{time}}$.

### C.6.1 Neighborhoods

Given a solution, let $\bar{B}$ and $\bar{D}$ be the set of blocks and duties in the solution. Three neighborhoods are defined for the E-VCSP and are as follows:

1. Random removal of duties and repair CSP
   The neighborhood is defined by randomly removing a set of duties from the current solution and repairing it using the heuristic B&P method for the CSP. Let $D^R$ denote the set of removed duties. The degree of destruction parameter $\xi_1$ controls the number of duties to be removed, which is determined as $|D^R| = \xi_1 \cdot |\bar{D}|$. After the removal of duties, $\bar{D}$ is updated as $\bar{D} = \bar{D} \setminus D^R$. The duties in the destroyed solution remain fixed in the
B&P setting, i.e. \( x_d = 1 \ \forall d \in \tilde{D} \). Additionally, to speed up the solution process, we use an early termination criterion in the B&P heuristic. The column generation algorithm at each node of the B&B tree is terminated if the LP objective does not improve by 0.001% in the last 10 iterations. We refer to this neighborhood as n-CSP.

2. Random removal and repair by sequential approach
The current solution is destroyed by randomly removing blocks and their corresponding duties. The destroyed solution is repaired by a sequential approach, where the E-VSP is repaired first and then the CSP. \( B^R \) denotes the set of removed blocks. The number of blocks to be removed, \(| B^R |\), is controlled by the parameter \( \xi_2 \) and is determined as \(| B^R | = \xi_2 \cdot | \tilde{B} |\). Trips, deadheads and idle times associated with the removed blocks need to be determined in order to remove the duties. The set of trips to be removed from the solution is represented as \( T^R = \{ t \mid a^2_{td} = 1, \ t \in T, \ b \in B^R \} \). The set of deadheads to be removed from the solution is represented as \( F^R = \{ f \mid a^3_{fb} = 1, \ f \in F, \ b \in B^R \} \). Similarly, the set of idle times to be removed is represented as \( I^R = \{ i \mid a^6_{id} = 1, \ i \in I, \ b \in B^R \} \). The E-VSP and the CSP are repaired using the heuristic B&P method described in Section C.5.3 and the variables in the solution remain fixed in their respective problems, i.e. \( y_b = 1 \ \forall b \in \tilde{B} \) and \( x_d = 1 \ \forall d \in \tilde{D} \). Similar to n-CSP, the early termination criterion is used in the B&P setting for both the E-VSP and the CSP. This neighborhood is referred to as n-Sequential.

3. Worst (and random) removal and repair by integrated approach
Let \( Dur_t \) denote the duration of trip \( t \in T \) in minutes, which is calculated from the departure and arrival time of \( t \). A function \( \delta_d \) is used to evaluate the duties in solution and \( \delta_d \) is determined as \( \frac{c^d_d}{\sum_{t \in T} a^2_{td} \cdot Dur_t} \ \forall d \in \tilde{D} \).

Similarly, function \( \Delta_b \) is used to evaluate blocks in the solution and \( \Delta_b \) is determined as \( \frac{c^b_b}{\sum_{t \in T} a^1_{tb} \cdot Dist_t} \ \forall b \in \tilde{B} \), where \( Dist_t \) is the distance of trip \( t \in T \) in km. Since a fixed and variable cost is associated with blocks and duties, it is preferable that the blocks and duties in the solution are efficient. A high value of \( \Delta \) and \( \delta \) indicates the inefficiencies of blocks and duties with respect to the amount of distance and time spent in covering the timetabled trips. As part of the intensification strategy, some of the inefficient blocks and duties are considered to be removed from the solution. The parameter \( \xi_3 \) controls the degree of worst removal and the
removal operation is carried out in the following three steps,

- Initially, a duty candidate list of size $\xi_3 \cdot |\bar{D}|$ is created by selecting duties in the descending order of $\delta_d \ \forall d \in \bar{D}$. A random duty $d^c$ is selected from the candidate list and added to the set of duties to be removed, $D^R$. Blocks that are associated with duty $d^c$ with respect to trips, deadheads and idle time are determined and added to the set of blocks to be removed, $B^R$. The blocks in the solution are updated as $\bar{B} = \bar{B} \setminus B^R$.

- Secondly, a block candidate list of size $\xi_3 \cdot |\bar{B}|$ is created by selecting blocks in the descending order of $\Delta_b \ \forall b \in \bar{B}$. A random block $b^c$ is selected from the candidate list and added to $B^R$.

- If $|B^R| < \xi_3 \cdot |\bar{B}|$, then random blocks are selected from $\bar{B}$ and added to $B^R$ until $|B^R| = \xi_3 \cdot |\bar{B}|$. The set of duties to be removed $D^R$ is updated based on $B^R$ as described in $n$-Sequential. Finally, $\bar{B}$ and $\bar{D}$ are updated as $\bar{B} = \bar{B} \setminus B^R$ and $\bar{D} = \bar{D} \setminus D^R$.

At the start of the heuristic, one may find many inefficient blocks and duties in the solution. The worst removal operation attempts to tackle such inefficiencies. However, during the course of the heuristic, further diversification strategies may be needed to reach unexplored parts of the solution space of the E-VCSP. Therefore, a pure random removal operation is proposed after the heuristic performs $\eta$ iterations. The removal operation is similar to that of $n$-Sequential and is controlled by parameter $\xi_3$. In both cases, the solution is repaired by the integrated approach described in Section C.5.1. A heuristic B&P method is devised to find integer solutions. A mixed branching rule that initially fixes block variables and then the duty variables is implemented. Similar to the other neighborhoods, the variables in the destroyed solution remain fixed, i.e. $y_b = 1 \ \forall b \in \bar{B}$ and $x_d = 1 \ \forall d \in \bar{D}$. One of the drawbacks of the integrated approach is that it is very time consuming. Hence, a time limit ($n_{time}$) is kept at every node of the B&B tree. This neighborhood is referred to as $n$-Integrated.

Figure C.5 shows the flowchart of the ALNS heuristic for solving the E-VCSP.
C.7 Computational Study

C.7.1 Instances

Three real-life instances are obtained from transport companies in Denmark and Sweden to test our algorithms. Table C.1 shows the three instances. *DK1* and *DK2* instances are from a transport company that operates in one of the largest cities in Denmark. *SE1* instance is from a transport company in Sweden that operates in both urban and extra-urban regions. Table C.1 also shows the E-vehicle and crew operational rules of the test instances. The crew operational rules for the *DK1* and *DK2* instances do not differ much. During operation in an extra-urban region, the vehicles are driven for an extended period. Therefore, the drivers are given longer breaks as indicated by the minimum break duration.

Figure C.5: Flowchart of the ALNS heuristic for solving the E-VCSP. The initial solution is denoted as $s^*$. At each iteration, a neighborhood is selected that destroys $s^*$ and repairs it by a heuristic B&P method. If the resulting solution is better, then it is saved as $s^*$ and the heuristic continues until the stopping criterion is met. The heuristic returns solution $s^*$ upon termination.
(i.e. 45 minutes) for the SE1 instance. Table C.2 shows the test instances that are categorized into sets of small, medium and large sized instances. The small and medium sized instances are extracted from the large instances DK1_3, DK2_5 and SE1_5. The table also gives an overview of the instances based on characteristics of trips, deadheads and idle times. For the SE1_5 instance, 48 timetabled trips are estimated to cover over 50 km each, which indicates operation of vehicles in an extra-urban region.

<table>
<thead>
<tr>
<th>Instance</th>
<th>E-vehicle operational rules</th>
<th>Crew operational rules</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max. distance without recharging (km)</td>
<td>Min. recharging duration (minutes)</td>
</tr>
<tr>
<td>DK1</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>DK2</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>SE1</td>
<td>120</td>
<td>600</td>
</tr>
</tbody>
</table>

Table C.1: E-vehicle and crew operational rules of test instances. DK1 and DK2 instances are from a transport company in Denmark, and SE1 instance is from transport company in Sweden.

<table>
<thead>
<tr>
<th>Category</th>
<th>Instance</th>
<th></th>
<th>Trips</th>
<th>Deadheads</th>
<th>Idle times</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Avg. distance (km)</td>
<td>Avg. duration (minutes)</td>
<td>Avg. distance (km)</td>
<td>Avg. duration (minutes)</td>
</tr>
<tr>
<td>Small</td>
<td>DK1_1</td>
<td>124</td>
<td>21.33</td>
<td>55.68</td>
<td>351</td>
</tr>
<tr>
<td></td>
<td>DK2_1</td>
<td>103</td>
<td>18.82</td>
<td>48.63</td>
<td>425</td>
</tr>
<tr>
<td></td>
<td>SE1_1</td>
<td>118</td>
<td>22.62</td>
<td>40.48</td>
<td>1,287</td>
</tr>
<tr>
<td></td>
<td>DK1_2</td>
<td>110</td>
<td>25.50</td>
<td>38.22</td>
<td>485</td>
</tr>
<tr>
<td></td>
<td>DK2_2</td>
<td>115</td>
<td>30.66</td>
<td>76.60</td>
<td>230</td>
</tr>
<tr>
<td></td>
<td>DK2_3</td>
<td>3</td>
<td>280</td>
<td>25.17</td>
<td>2,623</td>
</tr>
<tr>
<td></td>
<td>DK2_4</td>
<td>2</td>
<td>284</td>
<td>21.31</td>
<td>859</td>
</tr>
<tr>
<td></td>
<td>DK1_3</td>
<td>5</td>
<td>258</td>
<td>20.69</td>
<td>2,092</td>
</tr>
<tr>
<td></td>
<td>DK1_4</td>
<td>3</td>
<td>284</td>
<td>11.98</td>
<td>1,564</td>
</tr>
<tr>
<td>Medium</td>
<td>DK1_5</td>
<td>16</td>
<td>424</td>
<td>16.83</td>
<td>5,618</td>
</tr>
<tr>
<td></td>
<td>DK2_5</td>
<td>13</td>
<td>1,109</td>
<td>19.54</td>
<td>9,418</td>
</tr>
<tr>
<td></td>
<td>SE1_5</td>
<td>16</td>
<td>980</td>
<td>18.69</td>
<td>17,794</td>
</tr>
</tbody>
</table>

Table C.2: Overview of test instances. The number of lines is denoted as |L|, the number of trips is denoted as |T|, the number of deadheads is denoted as |F| and the number of idle times is denoted as |I|.

The following subsections detail the results of the independent, integrated, sequential approaches and ALNS heuristics. The solution methods use IBM ILOG CPLEX version 12.9.0 as the LP solver. All experiments are carried out on an Intel Xeon Processor E5-2680v2 @ 2.80 GHz with four cores and 64 GB memory.

C.7.2 Results of independent and integrated approaches

We first present the results of the independent approach as the method guarantees to find optimal LP solutions of the E-VSP and ICSP in a reasonable computation time. Table C.3 shows the results of the independent approach.
The total computation time spent on solving the master problem and subproblems are also reported in Table C.3. For the largest instances, the lower bound can be computed in less than 18 hours. On average, 56.03% of the total computation time of the independent approach is spent on solving the subproblem of the ICSP.

<table>
<thead>
<tr>
<th>Category</th>
<th>Instance</th>
<th>LP Objective</th>
<th>Master problem time (seconds)</th>
<th>Subproblem time (seconds)</th>
<th>Total computation time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-vehicle</td>
<td>Crew</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td>DK1_1</td>
<td>88,649.77</td>
<td>0.83</td>
<td>0.17</td>
<td>0.56</td>
</tr>
<tr>
<td></td>
<td>DK2_1</td>
<td>81,743.29</td>
<td>0.05</td>
<td>0.05</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td>SE1_1</td>
<td>104,081.49</td>
<td>0.60</td>
<td>0.33</td>
<td>0.44</td>
</tr>
<tr>
<td></td>
<td>SE1_2</td>
<td>77,905.37</td>
<td>0.52</td>
<td>0.41</td>
<td>0.28</td>
</tr>
<tr>
<td>Medium</td>
<td>DK1_2</td>
<td>141,131.57</td>
<td>55.07</td>
<td>7.41</td>
<td>81.12</td>
</tr>
<tr>
<td></td>
<td>DK2_3</td>
<td>212,374.47</td>
<td>22.88</td>
<td>1.45</td>
<td>6.83</td>
</tr>
<tr>
<td></td>
<td>DK2_4</td>
<td>184,956.04</td>
<td>41.25</td>
<td>2.93</td>
<td>32.27</td>
</tr>
<tr>
<td></td>
<td>SE1_3</td>
<td>134,675.02</td>
<td>76.76</td>
<td>6.00</td>
<td>107.10</td>
</tr>
<tr>
<td></td>
<td>SE1_4</td>
<td>154,372.86</td>
<td>26.00</td>
<td>3.39</td>
<td>14.59</td>
</tr>
<tr>
<td>Large</td>
<td>DK1_3</td>
<td>232,736.26</td>
<td>381.15</td>
<td>12.99</td>
<td>395.23</td>
</tr>
<tr>
<td></td>
<td>DK2_5</td>
<td>720,104.64</td>
<td>5,199.32</td>
<td>86.49</td>
<td>5,607.27</td>
</tr>
<tr>
<td></td>
<td>SE1_5</td>
<td>568,925.90</td>
<td>3,804.90</td>
<td>122.82</td>
<td>3,973.74</td>
</tr>
</tbody>
</table>

Table C.3: Lower bounds (\(Z_{\text{Independent}}\)) found by the independent approach.

<table>
<thead>
<tr>
<th>Category</th>
<th>Instance</th>
<th>LP Objective</th>
<th>Master problem time (seconds)</th>
<th>Subproblem time (seconds)</th>
<th>Improvement (%)</th>
<th>Total computation time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-vehicle</td>
<td>Crew</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td>DK1_1</td>
<td>90,131.67</td>
<td>46.48</td>
<td>1.20</td>
<td>15.54</td>
<td>1.67</td>
</tr>
<tr>
<td></td>
<td>DK2_1</td>
<td>84,560.04</td>
<td>30.51</td>
<td>0.34</td>
<td>9.51</td>
<td>3.45</td>
</tr>
<tr>
<td></td>
<td>DK2_2</td>
<td>138,872.32</td>
<td>6.56</td>
<td>0.14</td>
<td>1.46</td>
<td>4.00</td>
</tr>
<tr>
<td></td>
<td>SE1_1</td>
<td>113,533.45</td>
<td>351.78</td>
<td>3.03</td>
<td>140.17</td>
<td>9.08</td>
</tr>
<tr>
<td></td>
<td>SE1_2</td>
<td>83,171.45</td>
<td>74.96</td>
<td>0.99</td>
<td>15.30</td>
<td>6.76</td>
</tr>
<tr>
<td>Medium</td>
<td>DK2_3</td>
<td>210,721.81</td>
<td>505.84</td>
<td>9.80</td>
<td>143.58</td>
<td>3.46</td>
</tr>
<tr>
<td></td>
<td>DK2_4</td>
<td>190,930.11</td>
<td>3,076.02</td>
<td>109.51</td>
<td>2,745.40</td>
<td>3.23</td>
</tr>
<tr>
<td></td>
<td>SE1_3</td>
<td>136,852.74</td>
<td>45,849.58</td>
<td>785.35</td>
<td>38,680.86</td>
<td>1.62</td>
</tr>
<tr>
<td></td>
<td>SE1_4</td>
<td>162,530.66</td>
<td>2,955.49</td>
<td>73.18</td>
<td>2,210.62</td>
<td>5.28</td>
</tr>
</tbody>
</table>

Table C.4: Lower bounds (\(Z_{\text{Integrated}}\)) found by the integrated approach. **Improvement** in lower bound is calculated based on the lower bound provided by the independent approach (\(Z_{\text{Independent}}\)).

Table C.4 shows the lower bounds found by the integrated approach described in Section C.5.1. A maximum computation time of 172,800 seconds (48 hours) was set for the integrated approach. Preliminary experiments showed that the method was faster when the equality signs in Equations (C.2) - (C.5) were replaced by “\(\geq\)” signs. One should note that these changes are made only for the integrated approach and the equality signs are retained in the mathematical model for the repair method of n-Integrated. In this paper, we use the barrier method in CPLEX to solve LP problems. Solving the E-VCSP by the integrated approach is found to be computationally difficult and optimal LP solutions could not be found within the time limit for instances DK1_2, DK1_3, DK2_5 and SE1_5. Table C.4 reports only the results of the instances for which optimal LP
solutions could be found within the time limit. The integrated approach provides improved lower bounds when compared to that of the independent approach. The improvement in lower bound is calculated as \( \frac{Z_{\text{Integrated}} - Z_{\text{Independent}}}{Z_{\text{Independent}}} \times 100\% \). The average improvement is found to be 4.28%. The best known lower bound \( (Z_{LB}) \) for each instance is determined from Tables C.3 and C.4. Table C.4 also reports the total computation time spent on solving the master problem, E-vehicle and crew subproblems. It is found that, on average, 66.75% of the total computation time is spent on solving the master problem.

### C.7.3 Results of sequential approach

Table C.5 shows the results of the sequential approach. The heuristic B&P method for solving the E-VSP provides solutions with an average optimality gap of 1.98%. Similarly, for the CSP, the average optimality gap of solutions is found to be 0.17%. The table reports the overall solution value as the summation of feasible objective values of the E-VSP and CSP. The percentage gap of the sequential solution from the best known lower bound is calculated as \( \frac{Z_{\text{Sequential}} - Z_{LB}}{Z_{LB}} \times 100\% \), and the average gap is found to be 9.33%. For the large instances, the average gap is found to be 15.83% and this shows that there is potential for improvement by integration. The computation times are in the range of 48 minutes-7 hours for the large instances. Additionally, the feasible solutions indicate that, on average, the crew cost is 73.71% of the total operational cost.

<table>
<thead>
<tr>
<th>Category</th>
<th>Instance</th>
<th>Solution value</th>
<th>Number of E-vehicles</th>
<th>Number of drivers</th>
<th>Gap (%)</th>
<th>Total computation time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>DK1_1</td>
<td>96,224.65</td>
<td>14</td>
<td>25</td>
<td>6.76</td>
<td>5.29</td>
</tr>
<tr>
<td></td>
<td>DK2_1</td>
<td>87,027.46</td>
<td>21</td>
<td>20</td>
<td>2.92</td>
<td>1.08</td>
</tr>
<tr>
<td></td>
<td>DK2_2</td>
<td>144,883.17</td>
<td>26</td>
<td>37</td>
<td>4.33</td>
<td>1.05</td>
</tr>
<tr>
<td></td>
<td>SE1_1</td>
<td>118,902.39</td>
<td>32</td>
<td>31</td>
<td>4.73</td>
<td>2.32</td>
</tr>
<tr>
<td></td>
<td>SE1_2</td>
<td>86,507.16</td>
<td>17</td>
<td>26</td>
<td>4.01</td>
<td>2.11</td>
</tr>
<tr>
<td>Medium</td>
<td>DK1_2</td>
<td>165,414.72</td>
<td>20</td>
<td>47</td>
<td>17.21</td>
<td>467.10</td>
</tr>
<tr>
<td></td>
<td>DK2_3</td>
<td>231,912.99</td>
<td>33</td>
<td>55</td>
<td>5.55</td>
<td>88.07</td>
</tr>
<tr>
<td></td>
<td>DK2_4</td>
<td>202,697.14</td>
<td>29</td>
<td>49</td>
<td>6.16</td>
<td>223.57</td>
</tr>
<tr>
<td></td>
<td>SE1_3</td>
<td>153,238.66</td>
<td>20</td>
<td>43</td>
<td>11.97</td>
<td>385.62</td>
</tr>
<tr>
<td></td>
<td>SE1_4</td>
<td>179,130.01</td>
<td>27</td>
<td>52</td>
<td>10.21</td>
<td>92.54</td>
</tr>
<tr>
<td>Large</td>
<td>DK1_3</td>
<td>284,817.38</td>
<td>34</td>
<td>67</td>
<td>13.78</td>
<td>2,891.63</td>
</tr>
<tr>
<td></td>
<td>DK2_5</td>
<td>786,360.15</td>
<td>103</td>
<td>187</td>
<td>9.20</td>
<td>24,115.71</td>
</tr>
<tr>
<td></td>
<td>SE1_5</td>
<td>708,414.64</td>
<td>98</td>
<td>178</td>
<td>24.52</td>
<td>15,339.42</td>
</tr>
</tbody>
</table>

Table C.5: Results of the sequential approach. The overall solution value \( (Z_{\text{Sequential}}) \) is the summation of feasible objective values of the E-VSP and CSP. Gap represents the quality of the overall solution when compared to the best known lower bound \( (Z_{LB}) \).
C.7 Computational Study

<table>
<thead>
<tr>
<th>Category</th>
<th>Instance</th>
<th>Solution value</th>
<th>Number of E-vehicles</th>
<th>Number of drivers</th>
<th>Gap (%)</th>
<th>Total computation time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>DK1_1</td>
<td>101,603.65</td>
<td>14</td>
<td>31</td>
<td>12.73</td>
<td>1.64</td>
</tr>
<tr>
<td></td>
<td>DK2_2</td>
<td>91,746.68</td>
<td>22</td>
<td>22</td>
<td>8.50</td>
<td>0.78</td>
</tr>
<tr>
<td></td>
<td>SE1_1</td>
<td>131,767.82</td>
<td>40</td>
<td>36</td>
<td>16.06</td>
<td>3.22</td>
</tr>
<tr>
<td></td>
<td>SE1_2</td>
<td>94,425.96</td>
<td>18</td>
<td>33</td>
<td>13.77</td>
<td>1.45</td>
</tr>
<tr>
<td>Medium</td>
<td>DK1_2</td>
<td>170,124.54</td>
<td>20</td>
<td>50</td>
<td>20.54</td>
<td>74.34</td>
</tr>
<tr>
<td></td>
<td>DK2_3</td>
<td>241,709.36</td>
<td>36</td>
<td>63</td>
<td>10.60</td>
<td>32.42</td>
</tr>
<tr>
<td></td>
<td>DK2_4</td>
<td>219,297.97</td>
<td>31</td>
<td>60</td>
<td>14.66</td>
<td>40.85</td>
</tr>
<tr>
<td></td>
<td>SE1_3</td>
<td>161,351.47</td>
<td>22</td>
<td>50</td>
<td>17.90</td>
<td>38.60</td>
</tr>
<tr>
<td></td>
<td>SE1_4</td>
<td>212,224.86</td>
<td>40</td>
<td>77</td>
<td>30.58</td>
<td>11.88</td>
</tr>
<tr>
<td>Large</td>
<td>DK1_3</td>
<td>281,005.78</td>
<td>34</td>
<td>86</td>
<td>20.74</td>
<td>483.91</td>
</tr>
<tr>
<td></td>
<td>DK2_5</td>
<td>856,513.45</td>
<td>118</td>
<td>231</td>
<td>18.94</td>
<td>1,640.13</td>
</tr>
<tr>
<td></td>
<td>SE1_5</td>
<td>847,747.29</td>
<td>162</td>
<td>262</td>
<td>49.01</td>
<td>677.23</td>
</tr>
</tbody>
</table>

| Table C.6: Solutions obtained by applying a sequential approach for each individual line $l \in L$. Gap represents the quality of the solution when compared to the best known lower bound ($Z_{LB}$). |

As mentioned earlier in Section C.5.3, the sequential approach could be applied to each individual line $l \in L$ and Table C.6 reports the results of such an approach. Feasible solutions are found in short computation times; for the large instances, the computation times are found to be in the range of 8-28 minutes. However, the quality of the solutions is found to be low with an average gap of 18.51% from the best known lower bounds. For the large instances, the average gap is found to be around 30%.

### C.7.4 Experimental setup of ALNS

This section describes the experimental and parameter setup of the ALNS heuristic. Table C.7 shows the degree of destruction parameter values of the different neighborhoods that are set for each category. The time limit $n_{time}$ for the repair method in n-Integrated is set to 60 seconds. The destroy method in n-Integrated is changed to random removal from worst removal after the heuristic performs 1000 iterations, i.e $\eta = 1000$. Parameter $\mu$ is set to 25 iterations that describes the criterion for updating weights of the neighborhoods. The score parameters $\psi_1$ and $\psi_2$ are set to 25 and 0 respectively, and $\lambda$ is set to 0.1. Tolerance level $\epsilon$ is set to 0.01 that is used as a termination criterion of the ALNS heuristic.

In this paper, we perform two sets of experiments that are based on different initial solutions and they are as follows:
Table C.7: Degree of destruction parameter values. $\xi_1$, $\xi_2$ and $\xi_3$ correspond to neighborhoods n-CSP, n-Sequential and n-Integrated, respectively.

<table>
<thead>
<tr>
<th>Category</th>
<th>$\xi_1$</th>
<th>$\xi_2$</th>
<th>$\xi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>Medium</td>
<td>0.3</td>
<td>0.3</td>
<td>0.25</td>
</tr>
<tr>
<td>Large</td>
<td>0.3</td>
<td>0.2</td>
<td>0.1</td>
</tr>
</tbody>
</table>

1. Line solution
The line solutions shown in Table C.6 serve as an input to the ALNS heuristic. However, the initial solutions have large gaps from the best known lower bounds. For these experiments, the maximum computation time ($\text{max} \text{time}$) of the ALNS heuristic is set to 86,400 seconds (24 hours).

2. Sequential solution
Alternatively, the ALNS heuristic could be initialized with the solution provided by the sequential approach, which is known to be more time consuming. However, the ALNS heuristic starts with a relatively good solution. In this case, the maximum total computation time of the sequential approach and the ALNS heuristic together is set to 86,400 seconds (24 hours).

The aforementioned sets of experiments are performed in order to evaluate the behaviour of the ALNS heuristic when it is initialized with high and low quality solutions. On both experimental setups, the ALNS heuristic is run five times for each instance. The best and average results are reported from the five runs.

C.7.5 Results of ALNS

Table C.8 shows the best and average results of the ALNS heuristic when it is initialized with the line solution. As mentioned earlier in Section C.6, the solution obtained from the ALNS heuristic is denoted as $Z_{ALNS}$. The percentage gap of the solution from the best known lower bound is calculated as $\frac{Z_{ALNS} - Z_{LB}}{Z_{LB}} \times 100\%$, and the average gap is found to be 5.77%. The improvement provided by the heuristic when compared to the sequential approach is calculated as $\frac{Z_{Sequential} - Z_{ALNS}}{Z_{Sequential}} \times 100\%$, and the average improvement is found to be 3.22%. For the large instances, the improvements are in the range of 0.06-4.37% on average. All the large instances and two of the medium instances (DK1_2 and SE1_3) are terminated before all the weights of the neighborhoods could converge below the set tolerance level.
### Table C.8: Results of ALNS heuristic when it is initialized with the line solution.

The best and average solution values ($Z_{ALNS}$) are reported based on five runs of the ALNS heuristic. **Improvement** indicates the benefit of the ALNS heuristic when compared to the sequential approach ($Z_{Sequential}$). **Gap** represents the quality of the solution when compared to the best known lower bound ($Z_{LB}$). A maximum total computation time of 86,400 seconds (24 hours) is set for the ALNS heuristic.

<table>
<thead>
<tr>
<th>Category</th>
<th>Instance</th>
<th>Best</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Solution value</td>
<td>Improvement (%)</td>
</tr>
<tr>
<td>Small</td>
<td>DK1_1</td>
<td>92,827.11</td>
<td>3.53</td>
</tr>
<tr>
<td></td>
<td>DK2_1</td>
<td>85,796.23</td>
<td>1.41</td>
</tr>
<tr>
<td></td>
<td>DK2_2</td>
<td>140,700.72</td>
<td>2.89</td>
</tr>
<tr>
<td></td>
<td>SE1_1</td>
<td>114,784.70</td>
<td>3.46</td>
</tr>
<tr>
<td></td>
<td>SE1_2</td>
<td>85,436.81</td>
<td>1.24</td>
</tr>
<tr>
<td>Medium</td>
<td>DK1_2</td>
<td>153,770.30</td>
<td>7.04</td>
</tr>
<tr>
<td></td>
<td>DK2_3</td>
<td>223,445.19</td>
<td>3.65</td>
</tr>
<tr>
<td></td>
<td>DK2_4</td>
<td>197,134.05</td>
<td>2.74</td>
</tr>
<tr>
<td></td>
<td>SE1_3</td>
<td>144,237.48</td>
<td>5.87</td>
</tr>
<tr>
<td></td>
<td>SE1_4</td>
<td>783,040.35</td>
<td>0.42</td>
</tr>
<tr>
<td>Large</td>
<td>DK1_3</td>
<td>251,048.58</td>
<td>5.20</td>
</tr>
<tr>
<td></td>
<td>DK2_5</td>
<td>789,162.17</td>
<td>4.42</td>
</tr>
</tbody>
</table>

### Table C.9: Results of ALNS heuristic when it is initialized with the sequential solution.

The best and average solution values ($Z_{ALNS}$) are reported based on five runs of the ALNS heuristic. **Improvement** indicates the benefit of the ALNS heuristic when compared to the sequential approach ($Z_{Sequential}$). **Gap** represents the quality of the solution when compared to the best known lower bound ($Z_{LB}$). A maximum total computation time of 86,400 seconds (24 hours) is set for the ALNS heuristic.

<table>
<thead>
<tr>
<th>Category</th>
<th>Instance</th>
<th>Best</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Solution value</td>
<td>Improvement (%)</td>
</tr>
<tr>
<td>Small</td>
<td>DK1_1</td>
<td>93,668.10</td>
<td>2.66</td>
</tr>
<tr>
<td></td>
<td>DK2_1</td>
<td>84,888.99</td>
<td>2.46</td>
</tr>
<tr>
<td></td>
<td>DK2_2</td>
<td>140,835.72</td>
<td>2.79</td>
</tr>
<tr>
<td></td>
<td>SE1_1</td>
<td>113,651.18</td>
<td>4.42</td>
</tr>
<tr>
<td></td>
<td>SE1_2</td>
<td>84,183.52</td>
<td>1.96</td>
</tr>
<tr>
<td>Medium</td>
<td>DK1_2</td>
<td>155,212.10</td>
<td>6.17</td>
</tr>
<tr>
<td></td>
<td>DK2_3</td>
<td>222,806.35</td>
<td>3.93</td>
</tr>
<tr>
<td></td>
<td>DK2_4</td>
<td>195,571.05</td>
<td>3.52</td>
</tr>
<tr>
<td></td>
<td>SE1_3</td>
<td>143,068.31</td>
<td>6.64</td>
</tr>
<tr>
<td></td>
<td>SE1_4</td>
<td>167,468.24</td>
<td>6.51</td>
</tr>
<tr>
<td>Large</td>
<td>DK1_3</td>
<td>252,772.73</td>
<td>4.55</td>
</tr>
<tr>
<td></td>
<td>DK2_5</td>
<td>775,980.68</td>
<td>1.32</td>
</tr>
<tr>
<td></td>
<td>SE1_5</td>
<td>677,292.53</td>
<td>4.39</td>
</tr>
</tbody>
</table>
Table C.9 shows the best and average results of the ALNS heuristic when it is initialized with the sequential solution. The average gap is found to be 5.38% and the average improvement is found to be 3.58%. For the large instances, the improvements are in the range of 1.17-3.97% on average. When compared to the results in Table C.8, the largest improvements are found for DK2 and SE1 instances.

Figure C.6 illustrates the progress of the ALNS heuristic when it is initialized with line and sequential solutions for the large instances (DK1_3, DK2_5 and SE1_5). The best solutions of the aforementioned instances from Tables C.8 and C.9 are used as examples for representation of the ALNS heuristic in Figure C.6.

Figure C.6: Progress of the ALNS heuristic when initialized with line solution and sequential solution for instances a) DK1_3, b) DK2_5 and c) SE1_5. x-axis shows the solution and y-axis shows the computation time in hours. The best solutions from Tables C.8 and C.9 are used as examples for representation of the ALNS heuristic.

Figure C.7 compares the quality of the solutions provided by the sequential approach and the ALNS heuristic for all instances. Table C.10 summarizes the results of the ALNS heuristic based on the instances DK1, DK2 and SE1. For DK2 instances, the average improvements made by the ALNS heuristic are relatively small and the average gap (less than 3%) indicates that the potential for improving further is limited. It is believed that the benefits of using an inte-
Figure C.7: Comparison of results of the sequential approach and the ALNS heuristic when it is initialized with line and sequential solutions for a) small, b) medium and c) large instances. **Gap** represents the quality of the solution when compared to the best known lower bound ($Z_{LB}$), and the average gap is reported for the ALNS heuristic.

In an extra-urban transport system are much more significant than in an urban transport system. In an extra-urban region, drivers have less opportunity to be relieved from attending to a vehicle and may have to travel further between bus stops and depot for taking breaks or ending their respective duties. Gaffi and Nonato (1999) and Huisman et al. (2005) have primarily focused on applying an integrated approach for extra-urban transport systems due to their highly constrained nature with respect to crew scheduling. Another specialized study of the VCSP is the application of time windows for the timetabled trips that was briefly discussed in Section C.2. Kliewer et al. (2012) show that such an approach enables break possibilities between trips and provides further improvements. Therefore, the current structure of the timetabled trips may inherently have break opportunities for the drivers to some extent and hence, could have influenced the relatively small impact of the integrated approach on $DK2$ instances. However, significant improvements are realized for $DK1$ and $SE1$ instances. The average gaps of the aforementioned instances suggest that there is room for further improvement. Since the independent approach is used to evaluate the solutions for some of the instances, there is also an increasing need to develop alternate methods that provide stronger lower
bounds in reasonable computation time.

<table>
<thead>
<tr>
<th>Instance</th>
<th>ALNS initialized with line solution</th>
<th>ALNS initialized with sequential solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg. improvement (%)</td>
<td>Avg. gap (%)</td>
</tr>
<tr>
<td>DK1</td>
<td>4.79</td>
<td>7.14</td>
</tr>
<tr>
<td>DK2</td>
<td>1.72</td>
<td>3.82</td>
</tr>
<tr>
<td>SE1</td>
<td>3.77</td>
<td>6.89</td>
</tr>
</tbody>
</table>

Table C.10: Summary of the results of ALNS heuristic based on the instances DK1, DK2 and SE1.

C.7.6 Analysis of neighborhoods

Table C.11 summarizes the average performance of the neighborhoods for each category. It reports information such as the average number of times each neighborhood was selected and their average computation times. On average, neighborhood n-Integrated is selected the most number of times, but it is also known to be the most time consuming part of the heuristic. The performance of the neighborhoods during the course of the heuristic, on average, vary marginally for the different experimental setups. When the line solution is used for initialization, n-CSP and n-Sequential perform equally well as n-Integrated at the initial stages of the heuristic. This behaviour is not observable when the heuristic is initialized with the sequential solution. However, n-CSP and n-Sequential still tend to provide improvements during the course of the heuristic. Figures C.8 and C.9 illustrate examples of the performance of the different neighborhoods for instance SE1_5 when the ALNS heuristic is initialized with the line and sequential solutions, respectively.

<table>
<thead>
<tr>
<th>Category</th>
<th>Avg. number of iterations performed</th>
<th>Neighborhood</th>
<th>Avg. number of times selected</th>
<th>Avg. computation time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>1,990.50</td>
<td>n-CSP</td>
<td>273.06</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n-Sequential</td>
<td>370.20</td>
<td>0.08</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n-Integrated</td>
<td>1,347.24</td>
<td>2.82</td>
</tr>
<tr>
<td>Medium</td>
<td>2,556.66</td>
<td>n-CSP</td>
<td>215.16</td>
<td>0.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n-Sequential</td>
<td>487.08</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n-Integrated</td>
<td>1,854.42</td>
<td>47.45</td>
</tr>
<tr>
<td>Large</td>
<td>2,394.87</td>
<td>n-CSP</td>
<td>627.93</td>
<td>10.96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n-Sequential</td>
<td>684.97</td>
<td>16.66</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n-Integrated</td>
<td>1,081.97</td>
<td>60.05</td>
</tr>
</tbody>
</table>

Table C.11: Summary statistics of the neighborhoods in the ALNS heuristic.
Figure C.8: Performance of neighborhoods for instance \( SE1\_5 \) when the ALNS heuristic is initialized with the line solution. The iteration number is shown on \( x\)-axis and the weight of the neighborhoods is shown on \( y\)-axis.

Figure C.9: Performance of neighborhoods for instance \( SE1\_5 \) when the ALNS heuristic is initialized with the sequential solution. The iteration number is shown on \( x\)-axis and the weight of the neighborhoods is shown on \( y\)-axis.
C.7.7 Sensitivity analysis of electric bus technology

A sensitivity analysis is carried out to study the impact of the driving range of E-vehicles on the total operational cost. Therefore, the ALNS heuristic is tested with different values of the maximum distance without recharging parameter and the values range from 120 to 250 km. One of the issues that transport companies have to consider during the process of electrification is the selection of the type of electric buses to purchase. E-vehicles with larger batteries have a longer driving range but have a high purchasing cost. Pelletier et al. (2019) study the electric bus fleet transition problem that aims to offer a strategic guidance to transport companies in determining the most cost-effective investment plan during the years 2020-2050 by analyzing various types of electric vehicles and charging technologies. In our study, a detailed investment cost-analysis of different types of E-vehicles is not given and the fixed cost of the E-vehicles remain the same for the different driving ranges. However, the primary focus of this study is to present managerial insights into the operational cost based on the driving range of E-vehicles. Table C.12 shows the results for DK1 and SE1 instances. The maximum distance without recharging of 120 km is set as the base scenario and is used to calculate the improvements in total operational cost for each instance. Additionally, it is assumed that the E-vehicles are fully charged at the start of the day, and Table C.12 reports the average number of times E-vehicles are recharged during operations. The sequential approach is performed first to initiate the ALNS heuristic, and the maximum total computation time is set to 86,400 seconds (24 hours). For each value of the maximum distance without recharging, the ALNS heuristic is run five times and the average results are reported.

The results from Table C.12 clearly indicate that the total operational cost tends to decrease as the driving range of E-vehicles is increased. On average, the total operational cost decreases by 8.21% when the driving range is increased to 250 from 120 km. The largest improvements are realized when the driving range is increased to 150 from 120 km. The average cost reduction is found to be minimal (less than 1%) when the driving range of E-vehicles is changed from 200 to 250 km. With longer driving ranges, E-vehicles can cover more timetabled trips with less number of deadheads to the depot for recharging. As a consequence, the number of recharges per E-vehicle and the total distance covered by the E-vehicles are reduced. In all cases, we also see that the number of E-vehicles required is less than the requirements of the base scenario. Reduction in the frequencies of recharging and deadheading activities of E-vehicles have a direct impact on the crew schedule and the operational cost. Additionally, for some instances (SE1_1, SE1_2, SE1_4 and SE1_5), the average number of drivers needed is significantly less than the base scenario; the average number of drivers needed is decreased to 145 from 171 for SE1_5 instance. However, for
Table C.12: Results of different driving ranges (120, 150, 200 and 250 km) of E-vehicle for DK1 and SE1 instances. The average results are based on five runs and the ALNS heuristic is initialized with the sequential solution. The maximum distance without recharging of 120 km is set as the base scenario and is used to calculate the improvements in total operational cost for each instance.

DK1 instances, the average number of drivers remains the same for the different values of the maximum distance without recharging. SE1 instances include the operations in an extra-urban region and the improvements gained by increasing the driving range of E-vehicles are substantial. For the SE1_5 instance, the total operational cost is decreased by 16% on average when the driving range of E-vehicles is increased from 120 to 250 km. Such a result suggests that E-vehicles with longer driving range may be more beneficial for carrying out operations in extra-urban regions. In general, this study signifies the practical importance of electric bus technology and its impact on the operational efficiency of transport systems. Furthermore, this study shows that the number of drivers can be reduced with better batteries as it holds particularly for SE1 instances.

A study with different recharging times of E-vehicles could be seen as an ex-
tension of the sensitivity analysis. Kooten Niekerk et al. (2017) state that E-vehicles are recharged faster if the charging facilitates have larger energy capacities, which are known to be more expensive. However, such a study could give insight into the impact of fast charging technologies on the total operational cost. The proposed heuristic could also be seen as strategic tool to analyze various scenarios with different types of E-vehicles and charging technologies, which could potentially aid transport companies in making crucial investment decisions based on the operational requirements.

C.8 Conclusion

In this paper, we have introduced the E-VCSP that studies the impact of integrating vehicle and crew scheduling problems while considering the limited driving range of electric vehicles. An ALNS heuristic that utilizes B&P heuristic methods is proposed to solve the E-VCSP. The proposed methodology was tested on real-life instances from public transport companies in Denmark and Sweden. The sizes of the large instances varied from 424 to 1,109 timetabled trips. The heuristic approach provided evidence of improved efficiency of transport system when the electric vehicle and crew scheduling aspects are considered simultaneously. By comparing to the traditional sequential approach, the heuristic found improvements in the range of 1.17-4.37% on average for the large instances. Additionally, a sensitivity analysis of the electric bus technology was carried out to indicate its implications for the crew schedule and the total operational cost. The analysis showed that the operational cost decreases by 8.21% on average when the driving range of electric vehicles is increased to 250 from 120 km. The proposed heuristic can be used in an operational setting to find cost-efficient electric vehicle and crew schedules for a given charging infrastructure and type of electric vehicles. Furthermore, the heuristic could also be seen as a strategic tool for transport companies that supports them in making decisions such as investment in battery capacities of electric vehicles and charging infrastructure based on the operational requirements.

This paper also illustrated the computational difficulty of solving the E-VCSP by column generation, where optimal LP solutions could not be found for some instances within a time limit of 48 hours. Exploring exact methods to find lower bounds in reasonable computation times is seen as future area of research. Another possible research direction is to incorporate more features of the electric vehicle batteries such as the energy consumption, non-linear charging behaviour and partial recharges. For some charging systems, drivers may be required to attend to the vehicle when it is being recharged. Therefore, the E-VCSP can be extended to handle and study such scenarios.


