Airport Ground Staff Scheduling

Clausen, Tommy

Publication date:
2010

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

Airport Ground Staff Scheduling

PhD thesis 2.2011

DTU Management Engineering

Tommy Clausen
March 2011

DTU Management Engineering
Department of Management Engineering

Airport Ground Staff Scheduling

Tommy Clausen

Kgs. Lyngby, 2010

Dansk titel:
Planlægning af jordpersonel i lufthavne

Department of Management Engineering
Technical University of Denmark

Produktionstorvet, Building 424
DK-2800 Kgs. Lyngby, Denmark
Phone: +45 45 25 48 00, Fax: +45 45 25 48 05
info@man.dtu.dk
www.man.dtu.dk

Summary

Modern airports are centers of transportation that service a large number of aircraft and passengers every day. To facilitate this large volume of transportation, airports are subject to many logistical and decision problems that must continuously be solved to make sure each flight and passenger travels safely and efficiently through the airport.

When an aircraft lands, a significant number of tasks must be performed by different groups of ground crew, such as fueling, baggage handling and cleaning. These tasks must be complete before the aircraft is able to depart, as well as check-in and security services. These tasks are collectively known as ground handling, and are the major source of activity with airports.

The business environments of modern airports are becoming increasingly competitive, as both airports themselves and their ground handling operations are changing to private ownership. As airports are in competition to attract airline routes, efficient and reliable ground handling operations are imperative for the viability and continued growth of both airports and airlines. The increasing liberalization of the ground handling market prompts ground handling operators to increase cost effectiveness and deliver fast and reliable service.

This thesis presents models and algorithms for general optimization and decision problems arising within ground handling. The thesis contains an introductory part which provide an overview of the ground handling environment and reviews a series of optimization problems from the specific perspective of airport ground handling. In addition, the thesis contains five scientific papers, which consider specific optimization problems within ground handling in detail. The

considered problems range from generalized approaches to workforce planning, to highly detailed scheduling problems arising in the highly dynamic environment of airports.

Resumé

Moderne lufthavne er transportcentre som betjener et stort antal fly og passagerer dagligt. For at håndtere de store transportmængder er lufthavne underlagt mange logistiske problemer og beslutningsproblemer som løbende skal løses for at sikre at hvert fly og passager rejser sikkert og effektivt gennem lufthavnen.

Når et fly lander, skal et betydeligt antal opgaver udføres af forskellige grupper af jordpersonel, f.eks. tankning af brændstof, bagagehåndtering og rengøring. Opgaverne skal være udført før flyet kan lette igen. Opgaverne benævnes i fællesskab ground handling og er hovedkilden til aktivitet i lufthavne.

Forretningsmiljøet i moderne lufthavne bliver mere og mere konkurrencepræget, eftersom både lufthavnene og deres ground handling operationer skifter til privat ejerskab. Da lufthavnene er i indbyrdes konkurrence for at tiltrække ruter fra flyselskaber, er effektive og pålidelige ground handling-tjenester nødvendige for at sikre rentabilitet og vækst for både lufthavne og flyselskaber. Den stigende privatisering på ground handling-markedet driver virksomhederne til at øge effektiviteten og levere hurtige og driftsikre ydelser.

Denne afhandling præsenterer modeller og algoritmer for generelle optimerings- og beslutningsproblemer der opstår indenfor ground handling. Afhandlingen indeholder en indledende del, som giver et overblik over ground handling og behandler en række optimeringsproblemer fra et konkret ground handling-perspektiv. Derudover indeholder afhandlingen fem videnskabelige artikler som betragter konkrete optimeringsproblemer indenfor ground handling. Problemerne strækker sig fra generelle tilgange til planlægning af arbejdsstyrken, til detaljerede problemer indenfor opgaveallokering der opstår i lufthavnens dynamiske omgivelser.

Preface

The work presented in this dissertation constitute in part the fulfillment of the requirements for acquiring the degree of Ph.D. at DTU Management Engineering as well as the "Industrial PhD diploma" awarded by the Danish Ministry of Science, Technology and Innovation for Ph.D. students following the Industrial Phd program.

The thesis was written in part at the Department of Computer Science, University of Copenhagen (DIKU) from February 2007 to February 2009 and later at the Section of Operations Research, DTU Management Engineering, Technical University of Denmark until its completion in August 2010.

As part of the Industrial Phd program, I have been employed at the privately held Danish company WorkBridge A/S, where I have spent 50% of my time during this study. During the time at WorkBridge, I have worked as an integral part of the development section, where the research results presented herein has been implemented into WorkBridge products. The majority of the work presented in this thesis is currently in operation at WorkBridge, or at customers worldwide.

The thesis deals with different aspects of optimization arising within airport ground handling. The thesis consists of an introductory part outlining the problems occurring within the field and a collection of five research papers. Of the papers, four are motivated directly from real-life applications within the field and two of them describe projects which are currently in operation.

One of the five papers is published and four are currently submitted to peer-reviewed journals within the field.

Kgs. Lyngby, August 2010

Tommy Clausen

Acknowledgments

This thesis could not have been written without a large number of people, to whom I owe a great deal of thanks.

First, the conception of this thesis is owed mainly to Erik Sørensen, who generously accepted to sponsor the project. In the same vein, I would like to thank my industrial supervisor Janus Sejr Jensen for helpful and dedicated supervision throughout the project, and Nicolai Graff Andersen for providing both myself and this project an integrated role within the WorkBridge development section. A general thanks to all at WorkBridge for your kind cooperation and many interesting discussions. A specific mention should go to the "algorithms team": Sara Bisander Nielsen, Thomas Gerken, Oliver Grandvuinet and Morten Nielsen, as well as the many other contributors to the knowledge and algorithms presented in this thesis.

At the university, I give a heartfelt thanks to my supervisor David Pisinger for excellent supervision and for bringing academic perspectives to a project that at many times has been anchored more deeply in industry than at the university. A particular thanks as well for allowing me to join the move to a new department at DTU, and a thanks to the entire Operations Research Section at DTU Management for welcoming me into the group.

Special thanks go to the co-authors and readers who have contributed directly to the written parts of this thesis: David Pisinger, to whom I owe much for the collaboration and review on many projects. Allan Nordlunde Hjorth and Morten Nielsen for cooperation on the long-lived Seat Reservation project. Line Blander Reinhardt for for invaluable work and dedication on the PRM project.

Torben Barth, Mette Gamst, Berit Løfstedt, Christian Erdinger Munk Plum and Bo Vaaben for the many helpful comments and discussions that arose from your kind reviewing sessions.

Finally, I am grateful to my girlfriend Maya, as well as family members and friends, who have endured the hardships of this projects with me and enjoyed none of the joys and excitements in return.

Contents

Part I

Airport Ground Handling

Chapter 1

Introduction

This thesis has been written in part at the Danish company WorkBridge A/S. The company provides software planning and scheduling solutions for resource management of mobile workforces. The WorkBridge product line includes automated planning and optimization systems for long-term planning, rostering, and real-time workforce scheduling, among many others. Within the field of airport ground handling, the WorkBridge systems are used by customers in airports worldwide.

At the time of writing, WorkBridge employs around 30 people, with the majority of the employees situated at the head office in Copenhagen, Denmark.

The work presented in this thesis is divided into two parts. The first part (which includes this introduction), describes airport ground handling and some of the optimization problems within the area. Despite the physically confined airport space, the variety of optimization problems and solution methods is extensive. To limit the extent of the presentation, I have chosen to use my experiences at WorkBridge as a vantage point and focus on broad issues in planning and implementation, rather than a methodical enumeration of optimization methods and models.

The second part of the thesis contains five scientific papers that reflect specific projects I have worked on during the time of this study. As such, the papers

should not be viewed as a selection that broadly covers the field of optimization in airport ground handling, but rather a collection of *selected topics* determined by a three year window in a lengthy and ongoing process of developing the WorkBridge planning systems.

As a result of the cooperation, most of the work presented in this thesis is performed on real-life optimization problems defined through discussions with WorkBridge staff as well as workshops with customers. The majority of the work presented herein is currently in use.

1.1 Motivation

The aviation industry is undergoing continuous development as air traffic becomes an increasingly important factor in our society, for both business and leisure. As a result, airports form a pivotal part of the infrastructure and economy of any population center. In 2004, 200,000 jobs were directly supported by aviation in the UK, and up to 600,000 jobs were supported by aviation indirectly [9].

Most airports are extremely busy environments which are operating at the peak of their capacity. Congestion in airports and in the airspace cause frequent delays, which put additional strain on already tight schedules. While many airports are in the process of expanding, constructing new airport facilities is slow and expensive. As such, short and medium term growth in the aviation sector depends on the airports to operate at high efficiency.

Airports are increasingly liberalizing their ground handling operations. Ground handling operations that have traditionally been subsidized to corporate divisions of the airport or the national airline are now being offered in free competition. In the European Union, the liberalization of the ground handling market by directive 96/67/EC has caused a rise in the number of third party ground handling companies of more than 80% between 1996 and 2007 [27]. This increased competition has caused increased need for effectiveness and cost minimization.

The emergence of low-cost carriers in recent years causes new challenges for ground handling operations. In contrast to traditional carriers, low cost carriers operate with very short ground times to minimize expenditures. This causes a highly constrained working environment for ground handling companies and an increased risk of disruption in case of delays.

Finally, recent world events have shown that aviation is particularly vulnerable

to outside influences. The economic crisis of 2008/2009 has had a severe effect on what was previously a booming economy. Although recent forecasts indicate that passenger levels are rising again [62], the crisis has caused a level of cost awareness that will undoubtedly remain in the coming years.

For airlines, crew expenditures account for a large amount of the airline's total expenses. Airlines have therefore devoted large resources to crew scheduling, and as a result airline crew scheduling has been an important research area in operations research for some years (see e.g. [60]). According to Herbers [57], the area of airport ground staff scheduling has in comparison received much less attention.

Crew for ground handling operations are typically less expensive than airline crew (e.g. pilots), where salaries can be very high. Also, since ground crews are located at the airport at all times, it is possible to call in extra hands if needed. This has made the need for a good planning solution less pressing for ground handling companies than airlines in the past. However with increasing cost awareness and more constrained working conditions, the need for automated planning aids is becoming significant.

1.2 Optimization in Real-Life Applications

When introducing planning systems into an organization that currently uses manual planning, there are large savings to be made. The direct, and most often quoted, saving is the minimization of resources obtained by producing more efficient schedules. Such savings have the advantage of being easy to communicate, but not all parts of the receiving organization may like the message. As pointed out by Brusco *"for purposes of employee relations, it is not advantageous to associate software enhancements with employee reductions"* [23].

Introducing planning systems within a company also has less visible benefits which may be just as important as cost minimization. Increasing the quality of schedules does not necessarily mean resource minimization. Better utilization of existing staff may lead to higher service levels, a better ability to absorb changes due to robustness and a higher level of fairness which leads to more employee satisfaction. Furthermore, automated planning can be used to create more uniform schedules to ensure a consistent level of service on different days.

The large number of rules and regulations inherent to many service organizations impose the same difficulties for manual planners as for optimization systems. In some cases it may be impossible to satisfy all constraints, in which case

optimization systems can be used to quickly examine different scenarios to find the least undesired solution. The fast validation of feasibility is an advantage that can save a lot of time. Bechtold [14] reports that the introduction of a computer-based planning system saved the planning staff a day or more each week, which can be used for other work than manually creating schedules.

When designing optimization systems for use in real-life applications, the models must also satisfy the large set of constraints inherent to real-life workplaces. An additional difficulty is that not all such constraints can be made explicitly known or modeled. This may be due to simplifications of the model for the sake of simplicity or tractability. Most optimization problems are NP-hard even in simplified forms, so it is common to have to "cut some corners" to obtain models that can be solved in reasonable time on the large instances required in real-life applications. Also, faulty interfaces to information systems may be unable to present relevant information in time. Finally, some information is difficult to input into the model, because it is conveyed by other means of communication, such as telephone calls, written notices or simple conversations between planners and scheduled personnel. Such analogue means of communication is a particular problem in organizations which are not accustomed to working with computer systems. Another aspect of this problem is the so-called "tacit knowledge" that is not explicitly formulated or written down. In the study by Powell et al. [83], discrepancies between information in the workforce and information in the model may lead to *user noncompliance*, where users may opt not to follow the recommendations of the schedule, because the user possess more accurate information than the model. The ability to strengthen decision making for all levels of information is a key challenge for optimization and decision support systems.

In most real-life applications, the schedules provided by the algorithm are used as a decision support tool to augment and support the work of a manager or planner. If the schedules deviate from the old way of doing things in the organization, carrying out the plans in practice becomes more difficult. As a consequence, a significant part of modeling for real-life problems is to replicate the existing behavior of the organization.

In a study of introducing planning software at airport stations, Brusco [23] notes that an important requirement for planning personnel to accept the use of the software is fast running times of less than five minutes. Although the study is now more than ten years old, the attitude of manual planners is not likely to have changed considerably. We therefore consider fast running times a requirement.

1.3 Thesis Overview

This thesis is structured in two parts. Part I provide an overview of optimization problems occurring within the field of airport ground handling. Part II consists of five individual scientific papers related to the topic.

Part I is divided into several chapters. Chapter 2 introduces the ground handling and the role of ground handling service companies in modern airport operations. In Chapter 3 planning problems are presented using a general timeline. The planning problems arising in ground handling are presented in the context of the timeline. Chapter 4 considers demand modeling, as well as topics in assigning a workforce to demand. Chapter 5 introduces problems and models for workforce planning and scheduling. Chapter 6 presents operational and dynamic problems with a review of literature relevant for problems arising in airport ground handling. Finally, Chapter 7 present the papers that constitute the second part of the thesis and Chapter 8 contain the conclusions of the thesis and directions of further work.

Part II contains the five papers that constitute the major scientific contributions of this thesis. Paper A presents a construction heuristic for the shift design problem, in which shifts must be created to cover a given demand as well as possible. Paper B presents a local search algorithm for the shift design problem in a generalized context, where the objectives and constraints are specified using a varying set of rules. Paper C presents an algorithm for dynamically dispatching vehicles transporting baggage for transfering passengers. Paper D presents a two-stage algorithm for scheduling a heterogeneous fleet of service vehicles for transporting passengers with reduced mobility through an airport. Finally, Paper E presents a branch-and-bound algorithm for a variant of two-dimensional packing with applications in seat reservation.

The layout of the papers follow the style of the remainder of the thesis, but the papers are otherwise sought to be as close to their most recent form as possible.

Chapter 2

Introduction to Airport Ground Handling

Modern airports are centers of transportation that service a large number of aircraft and passengers every day. Statistics from London Heathrow (one of the world's busiest airports) report more than 450,000 annual flights transporting more than 65 million passengers [99], which is slightly less than 1300 flights and 185,000 passengers per day on average. To facilitate this large volume of transportation, airports are subject to many logistical and decision problems that must continuously be solved to make sure each flight and passenger travels safely and efficiently through the airport.

Airports are an important part of the infrastructure in modern society. In particular, airports combine two prevalent modes of transportation, land-based and air transportation. According to Ashford et al. [6], the successful operation of any airport must balance the interaction of air transportation's main components:

- The airport operator
- The airlines
- The passengers

In most airports, decisions are made in a hierarchical fashion, where infrastructure resources and flights are planned first, leaving the scheduling of ground handling tasks to react to decisions and related changes. This hierarchy can be the result of a separation of responsibilities following de-regulation, limitations in current planning capabilities or simply historical conventions. Nevertheless, the ground handling area is often in the position of having to react not only to individual disruptions but also to compound disruptions caused by a combination of many airport and airline decisions.

2.1 Airport Planning Problems

In this section, we review the layout of a typical airport and some of the planning problems which arise at different levels of the airport planning hierarchy.

Figure 2.1 illustrates a typical layout of a modern airport. Airports consist of one or several terminal building, which contain public and restricted passenger areas. Public areas, which are accessible from the street, holds check-in counters, service desks, and so on. Restricted areas are only avialable through security security areas and holds lounges, shopping facilities, and a series of *gates*, which passengers pass through to board the airport. Some terminals use *piers* to extend terminals and increase the number of gates. The walking distances from the center of a terminal (where the recreational areas are usually located) can be significant, particularly with the use of piers. From the gate, the passengers may be able to walk directly into the flight, if the gate is equipped with a walkway. If not, they must walk to the flight across the *stand* where the aircraft is parked. In some cases, the flight may be parked at a *remote* stand, which is not adjacent to the terminal. In this case, the passengers must be transported by bus to and from the gate. The combination of adjacent / remote stands and availability of walkways varies from airport to airport. Due to capacity problems, some major airports have a significant number of remote stands [101]. Finally, airports house a number of auxiliary facilities to accommodate baggage handling, airport security, crew quarters, cargo, maintenance, etc. These facilities may be located in terminal buildings or in separate structures within the airport grounds.

The airport's operating hours may vary as some airports or operations within the airport operate continuously in 24/7 operation [21], while others do not operate at night [90].

Scheduling the use of runways for aircrafts to and from the airports is modeled by the aircraft landing problem which schedules the order of landing aircraft [13, 102] and the related aircraft departure problem which considers departures [7].

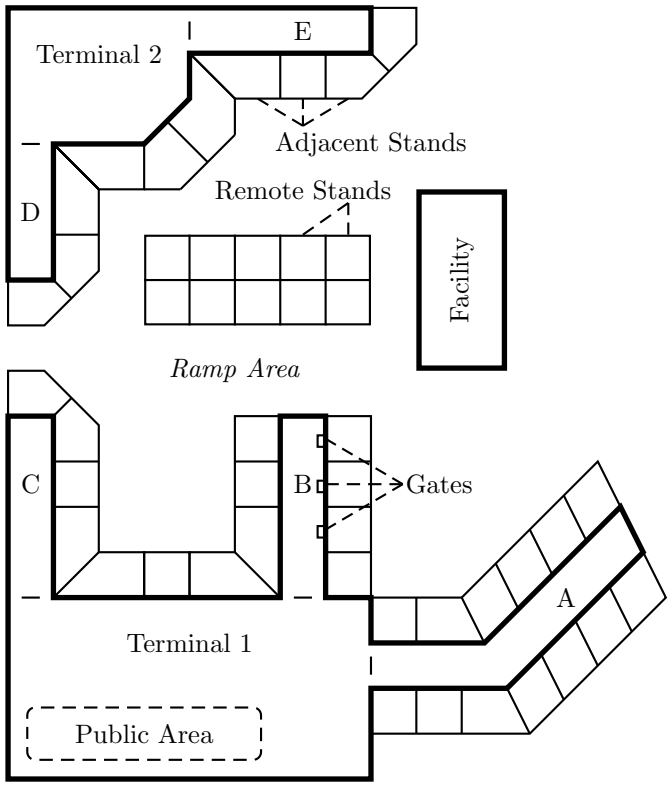


Figure 2.1: Schematic of airport with two terminal buildings T1 and T2, with piers A–E. The ramp area contains adjacent and remote stands, as well as auxiliary facility buildings. Public areas within terminals house check-in counters, service desks, etc.

The stand allocation problem determines stands for aircraft in order to minimize passenger walking distance [10, 56], to avoid congestion on taxiways [30] or to minimize the effects of delays in the form of waiting aircraft or relocations [43]. Other planning problems with focus on airport infrastructure include scheduling baggage conveyor belts [1] and the allocation of check-in counters [34, 47].

Common for all these problems are that they consider fixed airport resources, some of which are denoted *terminal resources* by Dorndorf [45], or *infrastructure resources*. As many airports operate at high capacity, efficient scheduling of these resources is increasingly important.

A second source of planning problems in airports is airlines. Each airlines flight schedules determine the activities at the airport, and are typically updated for each six month season. Minor updates to flight plans are made regularly and *re-timings* are done dynamically to minimize delays [67]. For airports that serve as airline bases, maintenance must be planned for the aircraft at regular intervals [3].

Some recent literature consider scheduling problems that cross the decision hierarchy by combining stand allocation with other areas of planning, such as workforce scheduling [90] and bus transportation [42]. In this text we consider ground handling from the perspective of a third-party service provider, which is not affiliated with the airport operator or a resident airline. For this reason, we do not include fixed terminal or infrastructure resources on our considerations.

2.2 Ground Handling

Ground handling is a common term to describe tasks that are performed at or around aircraft while the aircraft is on the ground, or otherwise relating to the aircraft's arrival or departure. This can include technical operations, such as refueling, or more service oriented work, such as check-in counter manning.

Ground handling work is commonly divided into *ramp* (or *apron*) tasks and terminal tasks. Figure 2.2 illustrate an overview of tasks occurring in ground handling, from the terminal and ramp perspectives. For an in-depth description of the tasks and processes involved in ground handling, see Ashford et al. [6].

Ramp tasks take place on the stands where aircraft are parked while they are on the ground. These tasks include handling the aircraft at the stand - pushback, power supply, etc. Other common ramp tasks include aircraft cleaning, toilet & water, catering, and baggage loading and de-loading, as well as transportation

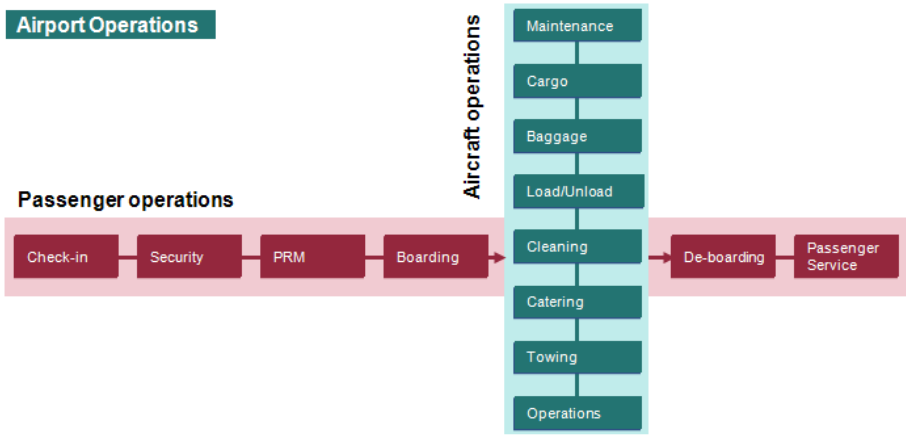


Figure 2.2: An illustration of tasks related to a flight. Terminal tasks (horizontal) are performed within the terminal for each passenger upon departure (left of column) and arrival (right of column). Ramp tasks (vertical) are performed while the aircraft is on the ground.

of passengers, baggage and cargo to and from the aircraft.

The teams performing the tasks will travel from task to task using vehicles. Major airports span a sizable geographical area that includes many stands and often multiple terminal buildings. For some examples of optimization problems relating to ramp work, see [31, 32, 59].

Terminal tasks are performed within the terminal buildings, and are usually centered around passenger service. Tasks in this category include manning check-in counters, boarding and de-boarding assistance. These tasks are performed within the terminals, in or adjacent to passenger areas. Examples of terminal work include [23] and [92]. Although tasks in passenger services are usually limited to the confines of a single terminal building, the travel distances can be significant [90].

Due to historical developments and economical consolidations, ground handling service providers are rarely small independent companies which provide a single service. Commonly, these divisions will handle multiple areas of service and be linked to similar divisions providing other services within the same airport or similar services at other airports. Ground handling operations are often handled by divisions of the airport management [42], one of the resident airline companies [88, 23, 76], or a large international service company [59].

The structure of subsidiaries and service companies is the result of de-regularization and privatization laws. Previously, many airports and airlines were state-owned, naturally causing the ground handling operations to be performed by a division of either the airport or the airline. More recently, the de-regularization has caused other service providers to enter the venue and it is common to have two or three service companies for many ground handling activities in the airport, each servicing a different set of airlines.

Most ground handling work is performed by service companies that are subsidized to handle the operations of one or more airlines at the airport. Each service company may offer a specific range of services, causing each airline to have several ground handling companies servicing the aircraft in various ways. For example, one company may handle check-in while another company handles refueling. Service operations are economies of scale, so ground handling companies will often provide several ground handling services to stay competitive. This may either be several types of services at the same airport, the same type of service at different airports, or both. See for example Brusco et al. [23] or Ho and Leung [59] for examples of planning in ground handling for multi-facility operations. To fully achieve the benefits of combining operations, it is essential to utilize cross-operational planning, particularly with similar operations within the same airport.

Chapter 3

The Planning Horizon

In this chapter we review a general model of planning problems and how they are linked with regard to the time of decision making and the level of detail they consider. In the final part of this chapter, we introduce a model of planning and decision making for ground handling operations and use the model to motivate the following chapters.

The planning timeline is introduced in Section 3.1 and the connection between the planning horizon and levels of detail is presented in Section 3.2. Finally, a planning horizon model for ground handling is presented in Section 3.3.

3.1 The Planning Timeline

Planning problems exist in a large variety, spanning many different industries and applications. Depending on the industry and application, the information available may vary. So may the complexity of carrying out the decisions made when solving the planning problem. Some planning problems may require the solution to another planning problem, dictating that some problems should be solved before others. As an example, consider two decisions that need to be made within a company:

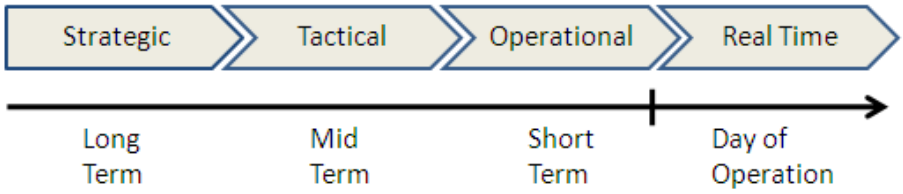


Figure 3.1: Planning timeline

1. What are the goals for the next ten years?

2. What should be done tomorrow?

These are very different questions that will have very different answers.

The first question asks the decision maker to generate a long-term strategy for the company. The second asks for a detailed plan for the following day. The problems operate on different levels of abstraction and complexity, and have different requirements to the time used to solve them. The long term question will probably need a series of meetings with top executives to formulate an answer.

The second question must be answered before the next day starts and it is unlikely that the top executives can spare the resources to contribute to the solution. Nevertheless, the two problems are related. The executives would expect the daily plans to adhere to the strategy they have formulated. Naturally, if this is to happen, the long-term problem should be solved before the short-term problem.

It is typical to classify planning problems according to the temporal dependencies between them. Long-term problems are denoted *strategic*, mid-term problems are denoted *tactical* and planning problems that deals with an actual day is denoted *operational* or *real-time*. For operational and real-time problems, the day they model is denoted the *Day of Operation*. Figure 3.1 illustrates the temporal dependencies of the problem categories and their comparative distance to the day of operation.

3.1.1 Strategic Planning

In this model, strategic planning occurs farthest from the day of operation and is mainly concerned with long-term decision making. We define for this purpose strategic planning as decision making that significantly modifies the size and composition of the workforce. Strategic applications include contract bidding, scenario analysis, and determining the workforce for a new operation. The demand modeling step is considered strategic when designing engagement standards and service levels for new operations or new contracts.

For ground handling operations, such decisions may include hiring or firing personnel, equipment acquisitions and scenario analysis for bidding on new contracts.

3.1.2 Tactical Planning

Tactical planning is concerned with planning problems closer to the day of operation. Problems in this category may generally be viewed as problems that determine availability on the day of operation from a pool of resources that has been fixed during the strategic planning phase.

Tactical planning considers planning for a workforce where the size is predetermined. This step is by some references referred to as *workforce scheduling* [48]. A typical scenario of tactical planning is to create rosters for a running operation that covers a future time period, such as next month or next season. In our model, tactical planning then covers rostering and may include parts of shift design and staff management. When rostering for a new season, an updated flight schedule can cause changes to the demand, which may require additional demand modeling and thus includes this step as well.

3.1.3 Operational Planning

Operational planning generates detailed execution plans for the day of operation. At this step, the demand and the resource availability is fixed. The problem is then to assign the work to individual resources as efficiently as possible under the conditions specified by the previous planning steps. This usually means performing as many tasks as possible with the available personnel, while ensuring that all operational constraints are satisfied.

Operational planning covers decisions specific to a day of operation, which in this context mainly covers daily planning problems in task scheduling. These may be calculated at the beginning of the day, or within a few days of the day of operation. Some staff management decisions may be viewed as operational, such as shift swaps, sickness and overtime handling.

3.1.4 Real-time Planning

In the final phase, real-time planning is concerned with adapting an existing plan for the day of operation to handle disruptions that may occur during the day.

Finally, real-time (or dynamic) planning reacts to events occurring during the day of operation, which in this model occur solely within task scheduling.

3.2 Planning and Levels of Detail

The planning timeline is a useful planning tool for several reasons. First, it illustrates the tendency that some decisions are naturally made before others, such as determining the size of the workforce (strategic planning), before determining how the workforce should be applied (tactical and operational planning).

A second useful aspect is the fact that reliable information becomes available at different times throughout the planning horizon. This tendency is of course most visible in real-time planning, where actual events deviates from their expected behavior. Another example is determining vacation and other absences such as training for individual workers. These decisions are typically made relatively close to the day of operation, in contrast to general decisions on managing the workforce, such as hiring or firing employees, as employee vacation wishes are collected.

As the day of operation approaches, more and more unknown factors become known. This increases the level of detail that should be modeled in the problems. The additional factors stem both from decisions arriving from external sources (such as vacation wishes), but also from an increased need to consider the context of the planning. As an example of this, consider the problem of designing weekend schedules for workers. Such a problem may be computed for a monthly planning horizon, but any restrictions on the placement of weekends may overlap with the previous planning period. If no two working weekends can

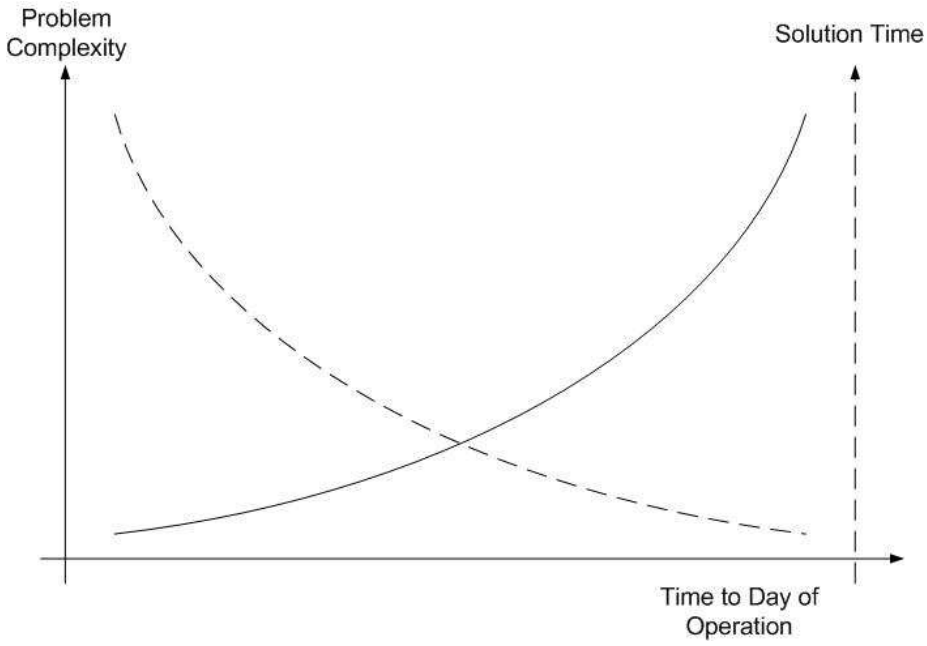


Figure 3.2: Expected trade-off between running time and level of detail. The level of detail increases as the day of operation approaches (solid axis and line), while the available computational time decreases (dashed axis and line).

be placed in a row, for instance, work in the final weekend of planning period $p-1$ and work in the first weekend of period p will violate this constraint. Such a constraint can only be expected to be satisfied if the context determined by the previous planning period is also considered. There is usually a direct trade-off between the available level of detail and the computational time available. This is illustrated in Figure 3.2.

In general, the implicit or explicit planning period will be shortened as the day of operation approaches. Decisions made in strategic planning may affect the organization for years to come, while planning for the day of operation will only affect the day itself.

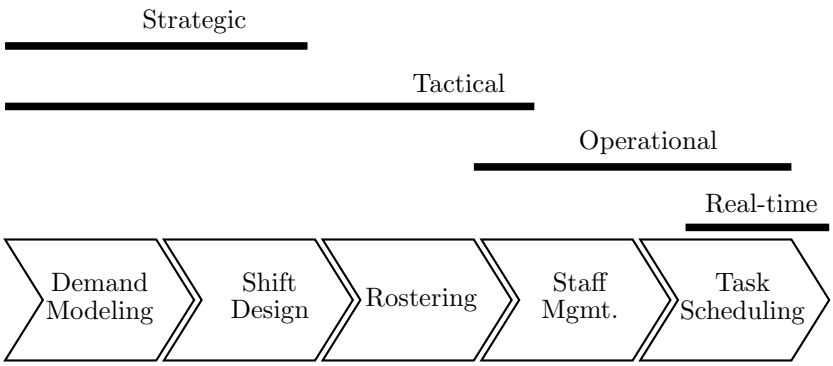


Figure 3.3: Problems in the ground handling planning model with references to the planning timeline. Demand Modeling and Shift Design may be viewed in a strategic or tactical context. Rostering is only considered for a predetermined workforce, which we consider tactical planning. Staff Management may be considered tactical or operational. Finally, Task Scheduling contains aspects of both operational and real-time planning.

3.3 A Ground Handling Planning Model

With reference to the planning timeline of Section 3.1, we can present a ground handling planning model, which conceptualizes the steps and planning flow of planning and scheduling in airport ground handling. The steps of such a planning model also reflect how and when different levels of detail are introduced to the planning process. The ground handling planning model is presented in Figure 3.3.

The planning model provides the basis for this presentation on workforce scheduling. It illustrates the specific steps observed within airport ground handling, and forms the basis of the review of specific problems in Chapters4.1, 5 and 6, as well as the papers introduced in Chapter 7. The different steps should not be considered as isolated problems. The output of one problem provides input to the subsequent problems. A a feedback loop may be imagined between each step, such that a step can be revisited to change the conditions for later steps. For example, if a problem is rendered infeasible, or if more robustness is desired.

The planning timeline division into strategic, tactical, operational and real-time planning can be combined with the decision-step based ground handling model as illustrated in Figure 3.3.

The main steps of the planning model are presented briefly in the sections below. We will focus on particular steps of the model, which are presented in detail in later sections.

3.3.1 Demand Modeling

Demand modeling is concerned with determining the amount of work that must be performed. Demand modeling is described in detail in Chapter 4.

3.3.2 Shift Design

Shift design is the problem of computing a set of shifts to cover the demand, which was determined in the previous step. The shift design step can be viewed as strategic when used to determine the minimum workforce size for an operation. When the size of the workforce is fixed, the characteristics of the workforce are added as constraints to the shift design problem, which can then be considered a tactical problem. We review shifts and the shift design problem in Section 5.2

3.3.3 Rostering

Rostering (or workforce scheduling) is the process of combining shifts into stretches of shifts and rest days to comply with labor regulations, fairness considerations, etc. We consider rostering in Section 5.3.

3.3.4 Staff Management

Staff management is the process of handling the work details of individual employees. The main part of this process is assigning employees to roster lines (thus detailing their work schedule), but may also include holiday considerations, the scheduling of training programs and shift bidding, sick leaves, etc. The area of staff management has not been an object of study in this thesis, so we will not consider it further. We refer to the survey by Ernst et al. [48] for a review of problems relating to staff management.

3.3.5 Task Scheduling

Task scheduling assigns tasks to shifts (which may in the broad sense include teams, equipment, or resources) for execution on the day of operation. Task scheduling include both operational and real-time problems. Both problem types are discussed in details in Chapter 6.

Chapter 4

Demand Modeling

In this chapter, we review aspects of the representation of work, or *demand*. An aggregated demand representation is reviewed in Section 4.1. Various types of demand are discussed in Section 4.2. The distribution of staffing is discussed in Section 4.3 and aspects of heterogeneous demand is presented in Section 4.4.

4.1 Aggregated Demand

Manpower planning uses an aggregated description of work that is used to make high-level decisions. For long-term planning in airports, it is common to model the work to perform using demand curves [46, 58]. An example demand curve for a single day is presented in Figure 4.1. For each time slot t an integer value d_t specifies the number workers required for the time slot.

These curves are commonly denoted *workload* or the *(forecast) demand*. The workload specifies the number of workers required during each time slot throughout the planning period. The demand curve representation is common in other applications in the service industry as well, such as call center manning [74, 81]

In airports, the majority of the demand is tied to the arrival or departure of flights, which leads the demand curves to retain traits of the airport's flight

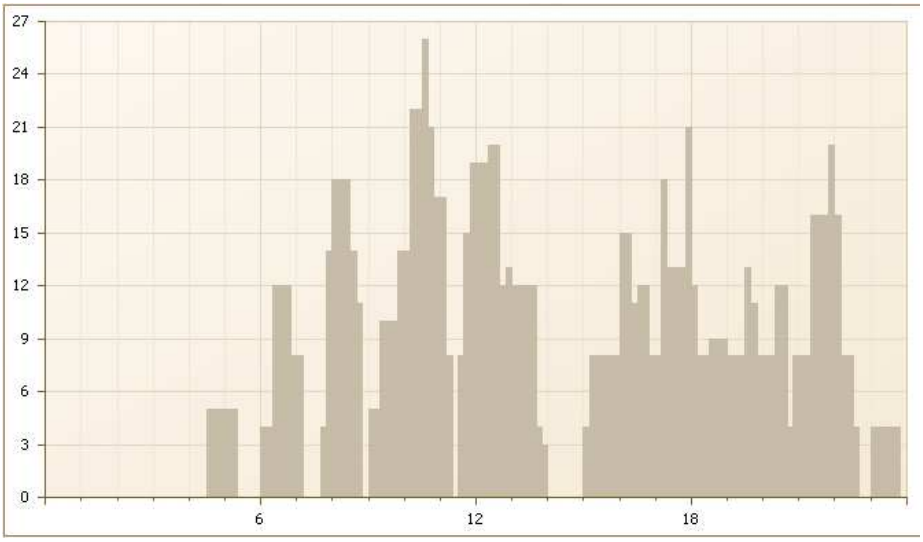


Figure 4.1: Example demand curve for a day of planning. For each time slot from 0 (midnight) to 24 (midnight) a number of required workers is specified

schedule. Many airlines have a tendency to group their flights into "banks" with many arrivals and departures within a short time, divided by intermediate periods of relative calm [67]. The banks cause the ground handling workload to fluctuate as well, causing large variations in staffing requirements from one time slot to the next [23].

More recent planning work in airports tends toward de-banking, i.e. spreading arrivals and departures more throughout the day [79]. The main focus of de-banking considerations are minimizing aircraft delays, which is another consequence of banked airline operations, but this will invariably have effect on ground handling operations as well.

For the purposes of the discussion in the remainder of this chapter, we consider the demand curve representation to be fixed as the method of representing work, as this is the representation that fit the proposed planning model of Section 3.3. Although this representation is widely used in ground handling and other service sectors, other approaches have been used as well. Herbers [58] propose a task-based workload representation and considers the shift design problem on the task-based workload [57]. Quimper and Rousseau [85] consider a shift scheduling problem that use a demand formulated on activities.

4.2 Demand Estimation

Models that use a demand curve representation of the work to fulfill, can in their basic form be presented as a variant of the original shift scheduling model proposed by Dantzig[39]:

$$\min \quad f(x) \tag{4.1}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} A_{it} x_i \geq d_t \qquad 1 \leq t \leq T \tag{4.2}$$

$$x_i \in \{0, 1\} \tag{4.3}$$

The model minimizes a cost function over the vector x of decision variables, where each x_i contributes to the demand d_t at time slot t if $x_i = 1$ and $A_{it} = 1$. A is a matrix of constant entries, $A \in \{0, 1\}^{n \times T}$. The demand d_t is integer.

From the perspective of the demand model (4.1)–(4.3), the problem of demand estimation is simple to formulate: Which value of d_t, $1 \leq t \leq T$ best describe the actual workload. The demands must be feasible, i.e. allow work schedules that cover the entire demand to perform all inherent work. Several demand representations can be feasible for the same amount of work and in this case it is desired to obtain a representation that can be covered with greatest efficiency, for example by containing the least units of demand, having the fewest fluctuations or satisfying some other property.

4.2.1 Sources of Demand

Estimation of the demand can be done in a variety of ways, which depend on the source of the demand. Some examples are given in the following.

Most ground handling work stems from tasks, which specify contiguous and isolated periods of work in a limited period of time. Tasks are usually obtained by combining the flight schedule with *engagement standards* which specify the type and volume of required work for each type of flight [58]. A collection of tasks may be converted into demand levels d_t by superimposing the tasks, i.e. adding all tasks covering each time unit.

For individual items which are handled identically, such as passengers and bags, it is common the use forecasts, which detail the expected number of items to arrive at different times. An example of this is the expected arrival times of passengers at check-in counters [92], which can be derived from the flight's departure time, destination, number of passengers, etc. Forecast can be derived

from advanced methods, such as queuing models. Lin et al. [74] use a combination of simulation and regression analysis of historical data to determine the demand for a call center. Mason et al. [77] use a greedy heuristic for minimizing demand for airport customs officers, while using simulation to ensure feasibility.

Static workload represents work that is predetermined by service contracts, rather than depend directly on passenger volume or flight movements. An example of this type of demand within an airport is the manning of service desk counters [23].

The main advantage of representing demand as individual d_t units is that demand stemming from all sources can be represented in this way. This means that it is easy to combine demand from different sources, or to use the same model for different operations. Using demand curves to represent task-based workload is an abstraction of the tasks considered in operational planning. This means that a number of dimensions of the underlying tasks cannot be modeled, including transportation times and time windows. We consider these special cases in the following.

4.2.2 Demand with Flexibility

When transforming work specifications into the demand curve representation, the essential decision is to decide which time slot should contain which unit of work in order to best represent the underlying demand. In the simplest terms, any demand can be added together to form a greater mass of workload. This works well when there is no flexibility in the underlying demand, such as when the demand is a set of stationary tasks or a forecast demands.

When the demand is more complex, some decisions must be made to create the workload, leading to the demand transformation becoming optimization or modeling problems in itself.

For tasks with time windows there is a flexibility in determining the execution time of the task. Transforming the demand into the demand curve, the flexibility and time windows disappear, so the placement of tasks will influence the subsequent planning problems. The problem of placing the tasks to generate a workload as smooth as possible is known as the resource leveling problem and is well-known from project planning [18].

Alfares and Bailey [5] present an integrated approach of project task scheduling and manpower scheduling where the workload profile can be changed by altering the start time, duration and worker assignments of each project task. A variant

of this can be observed in ground handling, when several close flights can be handled by the same crew by an optional or mandatory merging of tasks.

4.2.3 Demand from Transportation Problems

The planning and scheduling of personnel in transportation problems is well-studied in the literature. The scheduling and rostering of airline crew has in particular been subject to a high degree of interest, see for instance Hoffman and Padberg [60], Vance et al. [100] and the more recent survey by Kohl and Karisch [66]. Similar problems arise in the scheduling of railways crew [26] and bus drivers [104]. Common for these problems is that a task starts in one location and ends in another after a period of time, and both time and space considerations limit a crew members ability to perform other tasks. Demand in transportation problems are often considered as time/space networks. Scheduling and rostering problems for transportation crew are usually done by determining paths through the network, corresponding to the assignment of crew members to specific transportation legs.

In ground handling, many work types have an element of transportation as well, because work is centered around terminal or ramp locations, such as check-in counters, gates, aircraft stands, etc. As the allocation to these resources change throughout the day, the personnel travel between the locations to perform their tasks. An approximation of the transportation times can be modeled by extending each task with an average transport time. This may be sufficient when the transportation times are not significant, or if the times are largely similar.

If transportation times or other interdependent work characteristics are significant, determining feasibility is a bigger problem. A simple approach is to aggregate the demand into average capacities, such as "x demand items per hour", which by simple calculations can provide an approximate workload, which bounds the workload at each time period. This essentially considers transportation volume as a forecast workload.

An alternative is to use a simulation approach to determine a suitable demand. The simulation approach presented in Paper C of Part II for the baggage dispatch problem, could for example be extended to produce a demand curve.

4.3 Distributing Staffing

In the earliest references of shift scheduling, such as the set covering model (4.1)–(4.3), the demand per time unit was considered as constraints that must be met. Failure to meet the demand would result in an infeasible solution. The total cost of used shifts was minimized, thus implicitly minimizing surplus worker time units (*overstaffing*) as well. No care was given to the placement of the overstaffing, if any.

More recent references consider problems where the deviance of staffing from the demand is of interest. In these papers, the demand is sometimes denoted the *target demand* to emphasize that the specified levels are desired, rather than required.

In more recent literature, there are a number of references which deal with deviations from the target demand in different ways. Reasons for doing so include limitations on the availability [93], employee satisfaction in making all time slots "equally busy" [23], and increasing robustness by distribution the ability to absorb changes [76].

In some cases it may be desirable or unavoidable to consider *understaffing*, i.e. shortage of manpower, as well. This may be the case when the size of the workforce is fixed, as noted by Bailey [11]. Bailey proposed a model for this case, where understaffing is considered a "customer inconvenience" and receives a linear penalty depending on the time of the occurrences. Quimper and Rousseau [85] describe the cost of unperformed activities as *opportunity costs*, implying that it may be beneficial to avoid performing some activities.

Model (4.4)–(4.6) introduce a generalized model which extend the set covering model (4.1)–(4.3).

$$\min \quad \alpha_c \cdot f(x) + \alpha_u \cdot \Phi_u(u) + \alpha_o \cdot \Phi_o(o) \tag{4.4}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} A_{it} x_i + u_t - o_t = d_t \qquad 1 \leq t \leq T \tag{4.5}$$

$$x_i \in \{0, 1\} \tag{4.6}$$

The objective function (4.4) is a weighted sum of three terms, cost, understaffing and overstaffing, using weights α_c, α_u and α_o. The functions for understaffing $\Phi_u(u)$ and overstaffing $\Phi_o(o)$ is a generalized representation of the cost associated with the staffing deviances. To simplify the model, we use the vector vari-

ables $u = (u_1, \ldots, u_T)$ and $o = (o_1, \ldots, o_T)$ for understaffing and overstaffing in the objective. Constraint (4.5) balances the occurrence of shifts, understaffing u_t and overstaffing o_t with the demand d_t at each time unit $1 \leq t \leq T$.

The method of pricing understaffing and overstaffing represented by the Φ_u and Φ_o functions vary significantly in the literature. An overview of different approaches is given in Table 4.1. We describe some of the approaches in more detail in the following.

The simplest method is to penalize understaffing linearly, imposing an identical cost for each unit of understaffing and overstaffing. This approach, used by e.g. Thompson [95] and Musliu et al. [81] directs staffing levels to reach the target demand, but do not consider the distribution of understaffing or overstaffing across time.

Brusco and Johns [24] consider a two-stage approach that in the second step minimizes the maximum ratio of demand to overstaffing. Dowling [46] squares the overstaffing at each time unit. Chu [32] minimizes the maximum overstaffing.

Thompson [94] describes a model where different staffing levels can provide different levels of (acceptable) service, and thus generate varying profit levels, denoted Net Present Value (NPV). A unique cost is associated with each unit of overstaffing, reflecting the marginal profit of providing better service.

Keith [65] used a bounded and an unbounded variable for both overstaffing and understaffing (yielding four variables per time unit in total), where the bounded variables have lower penalty than the unbounded. The bounded variables create a "band" of acceptable (though still undesired) deviances from the ideal staffing levels.

Lusby et al. [76] adds an additional *contingency* demand, taken as the maximum of the previous time unit's demand and 15% of the time unit's original demand. The contingency demand can be left uncovered at a lesser penalty then the regular demand.

4.4 Heterogeneous Demand

In large organizations it is common for employees to be divided into different groups. If the groups are dissimilar enough, they can be considered in separate planning problems. However, there is often sufficient overlap between the employee groups to require that they are scheduled together.

Type	Example Reference	$\Phi_u(u)$	$\Phi_o(o)$
Constrained	Dantzig [39]	∞	-
	Lin [74]	$\begin{array}{ll} - & u_t < u'_t \\ \infty & u_t \geq u'_t \end{array}$	$\begin{array}{ll} - & o_t < o'_t \\ \infty & o_t \geq o'_t \end{array}$
Linear	Thompson [95]	$\sum_{t=1}^{T} u_t$	$\sum_{t=1}^{T} o_t$
Time-dependent	Bailey [11]	$\sum_{t=1}^{T} \Gamma(t) \cdot u_t$	-
NPV	Thompson [94]	∞	$-\sum_{t=1}^{T} \sum_{i=1}^{o_t} \Gamma(i,t)$
Banded	Keith [65]	$\begin{array}{ll} \sum_{t=1}^{T} u_t, & u_t \leq u'_t \\ \sum_{t=1}^{T} u_t, & u_t > u'_t \end{array}$	$\begin{array}{ll} \sum_{t=1}^{T} o_t, & o_t \leq o'_t \\ \sum_{t=1}^{T} o_t, & o_t > o'_t \end{array}$
	Lusby [76]	$\sum_{t=1}^{T} u_t$	$\begin{array}{ll} -\sum_{t=1}^{T} o_t, & o_t \leq o'_t \\ - & o_t > o'_t \end{array}$
Quadratic	Dowling [46]	$\sum_{t=1}^{T} (s_t - d_t)^2$	
	Paper B	$\sum_{t=1}^{T} u_t^2$	$\sum_{t=1}^{T} o_t^2$
Ratio	Brusco [24]	∞	$\sum_{t=1}^{T} \left(\frac{s_t}{d_t}\right)$
	McGinnis [78]	$\sum_{t=1}^{T} \left(\frac{s_t}{d_t}\right)^2$	-
Max	Chu [32]	∞	$\max_{1 \leq t \leq T} \{u_t\}$

Table 4.1: Examples of schemes for distributing deviances from the target demand d_t. The value s_t is the number of active shifts at time t and u_t and o_t are the units of overstaffing, respectively. Some models use preset limits o'_t and u'_t, or a function Γ, which depend on its parameters. When several formulas are given for the same type, different weights are implied. Formulas using o'_t and u'_t are weighted lower than their unbounded counterparts. Note that some formulas are negative, indicating a gain rather than a penalty. The value ∞ means that any amount is infeasible by a constraint and - means that amounts of the type is ignored.

In the most simple case of a heterogeneous workforce planning, employees are working under different contracts but are otherwise identical. This is commonly observed when the company employs both fulltime and part-time workers. The sets of constraints for different contracts can be completely parallel, such that all contract types have the same (full) set of constraints, but with different values, see e.g [20].

The capabilities of employees in the workforce may also differ. Due to seniority or different levels of training, not all employees are equally productive [96]. More experienced staff may be able to perform more advanced tasks in addition to the basic work. See e.g. [61] and [16] for further on the so-called hierarchical workforce problem.

More generally, the workforce may be multi-skilled [75] in which the workforce is divided into several types of capabilities. In the modelling of multi-skill workforces, the degree of overlap between skills in the workforce lie somewhere in between being completely separate (in which case the groups could be scheduled separately) and completely homogeneous.

To describe the level of cohesiveness between requirements $r \in R$ and qualifications (capabilities) $q \in Q$ we can use the *interaction level* int:

$$\text{int} = \sum_{q \in Q} \frac{|\{q \mid q \to r\}|}{|Q|}, \tag{4.7}$$

where $q \to r$ denotes that qualification $q \in Q$ can cover requirement $r \in R$. The interaction levels calculates the average number of requirements covered per qualification. The interaction level is introduced in Paper A.

A key difficulty in modeling technical personnel is that seemingly similar tasks may require many different combinations of workers or resources [17]. This is particularly true in airports, where different aircraft can require different equipment, or may require different forms of service, and different airlines can have unique requirements [59, 71, 91].

During check-in and boarding, some airlines require that check-in personnel wear the uniform of that particular airline. Computer terminals in check-in may vary from airline to airline, thus requiring specialized training. Specific language skills may also be a requirement for certain flights.

On the ramp, not all types of aircraft may be compatible with any type of vehicle or equipment. Thus, some lifts, loaders or pushback vehicles may only be able to service certain aircraft. For cleaning, different airlines will have contracted different levels of service for their fleet, or for specific flights. Certain parts of

the airport may be restricted and require special security access, that not all personnel will have.

4.4.1 Heterogeneous Coverage

When workload for different skill combinations exist, they are modeled using separate demand curves. Shifts may be able to cover work from more than one workload. When this is the case, the shift can cover different workload at each time period.

Calculating workload coverage may be seen as assigning segments of shifts to units of workload. As a shift is free to cover different workload at different time periods, the assignment can be made for each time period independently. For each time period, all shift segments must either be assigned a suitable unit of workload, or be assigned to cover nothing, which means overstaffing. Units of workload where no shift piece is assigned will not be performed and will figure as understaffing for workload of that skill.

In a homogeneous workforce, the assignment of (fixed) shifts, understaffing, and overstaffing to the demand is trivial, as it is simply the evaluation of the objective function of model (4.4)–(4.6). In a heterogeneous workforce the shifts (and thereby understaffing and overstaffing) can be allocated to different parts of workload. This means for an existing set of shifts, an assignment problem must be solved to find the mix between overstaffing and understaffing of different types. The heterogeneous workforce scheduling problem can be modeled by augmenting the homogeneous model (4.4)–(4.6) as follows

$$\min \quad \alpha_c \cdot f(x) + \alpha_u \cdot \Phi_u(U) + \alpha_o \cdot \Phi_o(O) \tag{4.8}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} A_{irt} x_i + U_{rt} - O_{qt} \geq D_{rt} \qquad \forall r \in R, 1 \leq t \leq T \tag{4.9}$$

$$\sum_{i=1}^{n} \sum_{r \in R} A_{irt} x_i + \sum_{r \in R} U_{rt} + \sum_{q \in Q} O_{qt} = \sum_{r \in R} D_{rt} \quad 1 \leq t \leq T \tag{4.10}$$

$$x_i \in \{0, 1\} \tag{4.11}$$

In comparison to the homogeneous model, the vectors d, u, and o has been replaced with matrices $D \in \mathbb{N}^{|R| \times T}$, $U \in \mathbb{N}^{|R| \times T}$ and $O \in \mathbb{N}^{|Q| \times T}$. The *requirements* $r \in R$ represent a minimum set of skills required to cover a type of demand. Demand and understaffing exist for each requirement r. *Qualifications* represent the combined skills of a worker following shift i. The overstaffing matrix O holds a separate value for each qualification. The objective function (4.8)

is a weighted sum of cost and the generalized understaffing and overstaffing functions Φ_u and Φ_o. It is identical to the objective function of the homogeneous model (4.4), except that the understaffing and overstaffing vectors have been replaced with two-dimensional matrices. Constraint (4.9) enforces staffing levels for each requirement and constraint (4.10) links the staffing levels across requirements and qualifications.

The problem of determining the right way of modeling deviances from the target demand is naturally more complicated than in the homogeneous case, which has had a number of different models, as seen in Table 4.1.

The simplest approach is to assign weights to each requirement and qualification and assign staffing to minimize weights. A simple observation, which is used in Papers A and B lends to the following hierarchy:

- If not all demand can be covered, the uncovered demand should be of the lowest priority. This can reflect that it does not (to the same degree) need to be covered, or that it will be easiest to find temp workers for this workload.
- Overstaffing should occur for the shift pieces with the highest priority. This allows key personnel to remain available for additional tasks of high priority.

The observations provide ideas to extend the majority of the modeling approaches of Table 4.1 to the heterogeneous case. For the quadratic approaches, an even distribution of staffing across requirements and qualifications may give a lower objective value than assigning hierarchically. This is also discussed in Paper B.

Chapter 5

Workforce Planning

The field of workforce planning in an organization is concerned with making sure the organization's workforce is capable of efficiently performing the work specified by service contracts. There are many aspects of workforce scheduling, ranging from large-scale decisions that affects the entire organization for years, to very detailed decisions on a very specific area of execution. Objectives of workforce planning problems include minimizing the cost of assigned workers, observing employee preferences or fairness, maximizing service levels, as well as many other considerations.

In this chapter, we give an overview of workforce planning and highlight some of the optimization problems that arise within the area. The overview is provided in Section 5.1 and in the following sections, we review individual steps of workforce planning in detail: The design of shifts is reviewed in Section 5.2. Workforce scheduling and rostering is considered in Section 5.3,

5.1 An overview of Manpower Planning

Manpower planning is concerned with determining work schedules for workers, in order to fulfill the demand. Following the time-dependent decision making

illustrated by the planning timeline, manpower planning can be subdivided into a number of subsequent decision or optimization problems. In many organizations, decision making for manpower planning is a step-wise process where each step covers higher levels of detail than the previous.

The subdivision of workforce scheduling problems has been noted by several influential papers in the literature. In the survey by Baker [12] three problems are cited: *Day off scheduling*, *shift scheduling* and *tour scheduling*. The first determines the days without work (or analogously the working days) for each employee. Shift scheduling determines a shift for each working day and tour scheduling is the combination of the two problems.

Tien and Kamiyama [97] present a more elaborate five-step procedure:

1. **Temporal Manpower Requirement.** This step determines the number of workers required at each time, by determining the type and number of shifts to follow.
2. **Total Manpower Requirement** determines the total workforce size required to meet the demand
3. **Recreation blocks** determines the size and numbers of contiguous blocks of days off, such as weekends.
4. **Recreation / Work Schedule** assigns the created recreation blocks to specific placements within the roster, thereby creating a day off schedule.
5. **Shift Schedule** Assigns shifts to each of the work days determined by the previous step, thereby producing a fully scheduled roster.

This subdivision illustrates the gradual introduction of details and decision variables into the decision making process. As also noted by Tien, the complexity of each step will vary from application to application. In this thesis, step 1 is covered by demand modeling, presented in Chapter 4, step 2 is covered by the shift design problem considered in Section 5.2 and in Papers A and B. The remaining steps are covered by Section 5.3.

A more recent survey by Ernst et al [48] for rostering problems subdivides workforce scheduling even further, into a set of *modules* where each application of manpower scheduling includes all or a subset of the modules.

A common message of the aforementioned surveys is that although all workforce planning and scheduling systems have common traits, the composition of a specific system depends on the application and must consider the organization and

operation it targets. Mason et al. [77] consider several modules in an application for staffing customs at an airport. Lin [74] describes a system for call center staffing which also consider several modules.

Musliu et al. [80] consider a four-step framework for general workforce scheduling, where each module generates a set of solutions which is passed to the next step. As each subsequent steps has a set of solutions as input, the schedules obtained at the end of the final step are improved.

5.2 Shifts

Shifts represent contiguous periods of work for a single worker, which start as the worker arrives at the workplace and end when the worker leaves for home. During the shift, a worker may perform tasks, move between tasks, hold breaks or be idle. It is preferred to have the worker perform tasks. Transport between tasks cannot be avoided and is part of the worker's day, but can be minimized by distributing tasks among workers in an efficient way. This is the main concern of task scheduling problems, which are described in Chapter 6. Time spent on breaks is mandatory and specified by labor regulations or union agreements.

The design and assignment of shifts can be subject to diverse set of constraints as well. Basic rules cover the minimum and maximum shift lengths, the number of breaks and their placement during a shift, minimum rest times between shifts and a minimum and maximum number of consecutive workings days allowed before the worker is entitled to one or more days of rest. Recent approaches use methods from constraint programming to model the many constraints inherent to workforce scheduling problems. See for example [37] and [85].

There can be a great variance in the flexibility of shift design. In some countries or organizations, employees in the service sector (and hence airports) can be subject to a large flexibility in their working hours, see e.g. [23] or [92]. A special case of flexibility in working hours is the handling of overtime, which may be paid as a supplementary salary or converted to additional rest time. The *annualized hours* considered by Azmat [8] may be considered an example of the latter, where the total number of working hours is fixed on a yearly basis.

Flexibility in starting times is usually considered across several days. A constrained set of starting time can replace the minimum rest time between shifts considered in more flexible applications. For example, Jacobs and Brusco consider a maximum variance on the start of shifts across consecutive days [64], as well as an airport staffing application where the number of different start-

ing times is limited and a minimum gap between starting times must be observed [20].

When designing shifts, a significant source of difficulty is in the flexibility of the shifts. The higher degree of flexibility, the more difficult the problems will typically be to solve. Introducing flexibility in starting times alone causes the number of shift combinations to grow rapidly [90]. Most shift scheduling problems therefore consider a small set of possible shifts to ensure that the problem is tractable [22].

The problem of creating shifts for the workforce to efficiently cover the demand is known as the shift design problem. It is a complicated problem that aims to cover the demand as efficiently as possible while observing the structure and many restrictions of the workforce. The shift design problem is investigated in Paper A where a dynamic programming heuristic is presented and in Paper B where an adaptive local search metaheuristic is presented that uses a flexible representation of workforce rules.

Other references have considered the shift design problem as well, using different methodologies. Musliu et al. [81] present a tabu search algorithm for the shift design problem. A tabu search algorithm is also used by Di Gaspero et al. to solve the *minimum shift design problem* [40] in which the number of different shifts should be minimized. Herbers [57] propose a branch-and-price approach to solve the shift design problem on tasks and an improvement algorithm using large neighborhood search based on constraint programming. The combination of local search and constraint programming is also used by Di Gaspero et al. for the combined shift and break design problem [41].

Workers are commonly entitled to a number of meal and relief breaks during a shift. If there is flexibility in the placement of breaks, the decision of when breaks should be held is often taken on the day of operation. It is not uncommon for breaks to contribute significantly to the duration of the shift, so in many operations the consideration of breaks is required for volume alone. The placement breaks can be subject to a complex set of constraints, such as feasible location of the breaks, minimum or maximum working time between breaks and settings where the number of breaks depends on the length of the shift. Complex constraints on the placement of breaks are considered by for example Rekik et al. [87] and in Paper A.

The *break scheduling problem* considers the placement of breaks within an exisiting set of shifts. See e.g. Beer et al. [15] and Widl and Musliu [103] for further details on break scheduling problems. A combination of shift and break scheduling is considered by Di Gaspero et al [41], using a hybrid local search and constraint programming approach.

5.3 Workforce Scheduling and Rostering

Workforce scheduling is concerned with the assignment of workers to specific working days and shifts, in order to cover the demand. Note that we generally assume the size of the workforce to be fixed, which is not assumed in all references.

In the original model by Dantzig [39], workforce scheduling is modeled using a set covering model:

$$\min \quad \sum_{i=1}^{n} c_i x_i \tag{5.1}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} A_{it} x_i \quad \geq d_t \quad 1 \leq t \leq T \tag{5.2}$$

$$x_i \in \{0, 1\} \tag{5.3}$$

The objective function (5.1) minimizes the cost c_i of each active shift, indicated by the decision variable x_i. Constraint (5.2) enforces that the demand d_t is covered at every time t in the planning horizon $1 \leq t \leq T$. The matrix entry A_{it} is 1 if shift i is active at time t and 0 otherwise. Finally, (5.3) enforces that all shifts are used at most once.

The basic model (5.1)– (5.3) can be extended in several ways. By setting $A_{it} = 0$ at times within shifts, breaks can be modeled. The planning horizon T can be extended to consider several days, in which each column can describe the activities of a worker through the planning horizon. In this way, constraints such as rest time between shifts and the placement of days off.

Historically, work on workforce scheduling can be divided into three groups [12]: (i) day off scheduling, (ii) shift scheduling and (iii) tour scheduling. Day off scheduling considers only the distribution of work days and days off. Shift scheduling determines the shifts to use on a single day. The model (5.1)–(5.3) is a shift scheduling model. Tour scheduling consider both problems in combination. Several surveys highlight workforce scheduling problems, for example Tien and Kamiyama [97] and Ernst et al. [48]. For tour scheduling in particular, see [4].

A limitation of the above model is that it grows rapidly with the size and detail level of the planning horizon, as well flexibility of the shifts. In particular the flexibility of breaks can cause a huge increase in the size of the model, as each shift must be represented in the model for any combination of break placements.

Several approaches have been taken to limit the size of the model. Some references use delayed column generation to iteratively introduce columns to the

model. Since only a few of the many columns are used in the final solution, this approach can significantly reduce the size of the model. See e.g. [57, 76]. Other methods to reduce the size of the model include implicit formulations, where the flexibility of shifts and (in particular) breaks is represented by special constraints, thus limiting the number of shifts. See e.g. [2, 21, 57, 87] for examples of this approach.

A closely related problem to workforce scheduling is *rostering* in which the goal is to produce a *roster* - a timetable of shifts and days off. Although the problems are largely synonymous and consider the same types of objectives and constraints, some circumstances are related directly to the roster.

Rosters are considered either named or anonymous. In named rosters, a specific employee is assigned to each roster line during the roster construction. This means that individual constraints and preferences can be taken into account. Named rostering is mainly considered within health care, such as *nurse rostering*[25].

In anonymous rostering, the employees are not assigned to roster lines until after the roster is complete. This means that personal preferences are not considered until after the rostering process, along with other personal considerations, such as vacation planning. This simplifies the rostering process, which can be necessary for organizations with a large workforce. Anonymous rostering is considered for most applications in the service industry, including airport ground handling [57].

A special case of anonymous rostering is the use of *cyclic rosters*. In these rosters, an employee does not follow a particular line in the roster, but moves to the next line when a line is complete. Cyclic rosters are inherently fair, since all employees will follow all lines. Although cyclic rosters are more complex than individual line rosters, a compact roster is valid for many weeks of operation. For examples of the use of cyclic rosters, see Brusco and Jacobs [19], Mason et al. [77], and Felici and Gentile [50].

Chapter 6

Task Scheduling

In this chapter we review problems relating to task scheduling, where the demand is considered as individual tasks with a high level of detail. Task scheduling is the set of problems dedicated to assigning tasks to shifts. Ground handling problems in this category are often solved close to the operation, and thus constitute the final steps of the planning timeline of Section 3.1: Operational problems and real-time problems.

Operational problems are concerned with the calculation of work schedules on a specific *Day of Operation*, where the actual work takes place. To achieve the maximum amount of detailed information, the operational problems are calculated as late as possible, often at the beginning at the day of operation, or at the end of the preceding day.

The related group of real-time problems is concerned with reacting to changes that occur during the day of operation, by updating the existing schedules.

In the remainder of the chapter, we review the specific problems in further detail. The modeling of the elements in task scheduling problems is presented in Section 6.1. Aspects of operational planning in ground handling are reviewed in Section 6.2 and real-time planning is reviewed in Section 6.3.

6.1 Modeling

In this section, we review the *building blocks* of the task scheduling problems: Locations, Resources, tasks, and skills.

A location l represents a physical area within the airport, such as a check-in counter, gate, aircraft stand or other area of importance to the operation. When performing work, it is often necessary to move from location to location during the day, which takes additional time. The time required to travel between two locations l_1 and l_2 is denoted the *travel distance*, $d(l_1, l_2)$.

A resource r represents a worker or a team of cooperating workers that is able to perform specific types of work. The resource has a *shift* that determines the start and end of the resource's work period. The shift of each resource is determined in previous planning steps, and is the result of *shift planning* or *rostering* optimization problems.

During the shift, the resource may have a number of breaks, where work cannot be performed. Associated with the shift are a location r_l where the resource must start and end the shift, as well as a location b_{rl} where the resource must start and end its breaks.

A task represents a contiguous block of work to be performed by a single resource (worker or team). A task t is represented by a start time s_t, duration d_t, and location l_t. If there is flexibility in the time of the task, this is represented by a time window (a_t, b_t) that the task must be performed within, i.e. that $a_t \leq s_t$ and $e_t \leq b_t$ must hold. Travelling from one task to the next will usually also mean changing location. When moving from location l_i to l_j, a travel time t_{ij} must be added to the schedule of the resource. Task t_j cannot be started after t_i is complete if $s_{t_j} < e_{t_i} + t_{ij}$. If there is flexibility in the times of the tasks, the tasks can be moved to accommodate the transport time, if permitted by the time windows. If there is no flexibility, t_i and t_j cannot be performed by the same resource.

A skill represents a singular capability of work that is associated with both resources and tasks. A resource can have a number of skills that specifiy the types of work the resource is able to perform. If the resource represents a team of workers, the skills reflect the types of work the team is able to perform in unison. Associated with a task is the set of skills required to perform the task. A task can only be performed by a resource, if all skills required for the task is posessed by the resource. The relationship between required (task) skills and posessed (resource) skills can be complicated, similar to the multi-skill considerations for workforce planning presented in Section 4.4.1.

6.1.1 Transportation Problems

Although task scheduling problems are more similar in time and scope than the class of manpower scheduling problems, there can still be a high degree of variation, originating either from general problem aspects or from the additional consideration of operational details.

A key factor in the general differentiation of task scheduling problems is the impact of transportation time. Some problems are mostly stationary problems, where the tasks are long and gathered within a small geographical area. This means only a small time of a shift's active time will be spent travelling between tasks, and the coordination required for travelling is reduced. Such problems can arise when manning check-in counters or service desks.

In other types of operations, the impact of traveling between tasks is greater. In these types of operations, the workers will spend more time traveling between tasks. Also, the tasks themselves are shorter, allowing the workers to perform more tasks during a shift, with more transport segments as well. Many ramp operations fall in this category, including aircraft cleaning, toilet & water services and catering. Problems in this category are similar to Vehicle Routing Problems in the literature. Solution methods to this kind of problem are typically inspired by this kind of literature.

Finally, other problems are almost purely transportation problems, where the stationary part of the work is minimal compared to the transport segments. This includes problems such as delivery of baggage, cargo, or transportation of passengers. These kinds of problems are naturally related to vehicle routing problems as well. Some problems are based on richer vehicle routing problems, such as pickup and delivery problems or multiple trip vehicle routing problems.

6.1.2 Mobile Workforce Scheduling

Many aspects of ground handling work contain a mix of transportation time between different locations and stationary work. Scheduling problems on this type of work is denoted *mobile workforce scheduling*. In airports, the geographical dispersion of tasks is limited. While the transportation times between tasks can be significant, they rarely exclude large enough portions of the remaining work to motivate using time/space networks used in the pure transportation problems, such as airline . In addition to airports, mobile workforce arise in the utility [53] and telecommunication [17] sectors. As noted by Borenstein et al [17], mobile workforce scheduling can be viewed as a combination of vehicle

routing and resource constrained project scheduling.

The literature of vehicle routing problems (VRP) is rich with examples of different application areas and operational constraints, see for instance Toth and Vigo [98] and Golden et al. [54] for references. The main concern of the VRP and variants is the performance of the scheduled fleet of vehicles, so the main objectives of these problems are minimizing the number of vehicles and the total travel length, as these are the direct contributions of the overall transportation cost.

In contrast, the resource constrained project scheduling problem (RCPSP) is concerned with the distributions of activities to resources such that all activities are performed as efficiently as possible while satisfying constraints on the precedence of activity completion and the capacities of individual resources. See e.g. [68] for a details on the resource constrained project scheduling problem. The most common objective for RCPSP is to minimize the *makespan*, i.e. the total time required to perform all activities.

A number of project scheduling variants have been considered in the literature, with varying objective functions and constraints. See e.g. [18] for an overview of project scheduling variants. Compared to vehicle routing, project scheduling is more concerned with the scheduling of tasks and do not consider transportation times. As noted by Borenstein et al. [17], the need of transportation considerations makes VRP the preferred reference for mobile workforce scheduling problems.

Cowling et al. [38] consider a mobile workforce scheduling problem as a generalization of the RCPSP in which some activities can be left unperformed.

Other problems related to mobile workforce scheduling include the *manpower allocation problem* in which workers mare assigned to jobs (or vice versa), and the *team orientation problem* in which team members cooperate to visit as many destinations as possible. For examples of the manpower allocation problem, see Chu and Lin [33], Lim et al. [73] or Li et al. [72]. For details of the team orienteering problem see Chao et al. [28].

6.2 Operational Optimization

For operational scheduling problems, most decisions relating to employees have already been made at the time of planning. The employees or teams available for scheduling are fixed to shifts where most parameters are fixed at the time of

calculation.

Operational ground handling problems are often viewed as steps in the overall planning process, rather than individual optimization problems. As such the goals of operational problems are rarely to minimize costs or resources but rather to execute daily plans under the conditions set by the solutions of the previous planning problems. Objectives for operational problems may include the even distribution of work throughout the day and across resources, and placing flexible breaks as advantageous as possible for the workers.

As unforeseen events may also occur from planning time to execution, it may not be possible to solve all tasks and the main objective is to schedule the resources to maximize the work solved.

In airports, a large diversity exists in the constraints imposed by the equipment, contractual obligations and work practices in different areas of ground handling work. Some examples are:

In the short baggage transportation problem considered in Paper C, each bag arriving in the airport and departing on another flight should be injected into the airport's regular (departure only) baggage handling process. This causes a transportation problem with a large number of bags, and where each bag can be delivered to a designated handling station if delivered early and must be delivered directly to the departing aircraft if close to departure.

In Paper D, a transportation problem is considered for passengers with reduced mobility, which must travel by several vehicles without being left unattended. This creates a large number of synchronization constraints to ensure that vehicles are present at the same time during each passenger hand-over.

Other references include Dohn et al. [44] who consider a mobile workforce scheduling problem for cleaning crews. On some flights, the aircraft is either very large or must be cleaned in a short period of time. These flights require a combination of crew who must coordinate their activities on the flight by beginning at the same time. Ho and Leung [59] present an algorithm for the scheduling of catering truck, which must coordinate the pickup and delivery of meal carts to the flights by either combining the pickup and delivery or splitting the flight for two trucks. In addition, the trucks can be manned by several combination of crew.

6.3 Real-Time Optimization

Real-time optimization consider problems in which changes occur during the day of operation. The schedule produced initially is naturally oblivious to the changes and cannot incorporate them directly.

By reacting to the changes during the day, the efficiency of the workforce can be maintained. For example by continuously scheduling tasks to ensure that the work is distributed fairly between workers. The process of updating schedules during the day is referred to as *re-planning*. We adopt the term *decision epoch* from Goto et al. [55] to describe a time instant where re-planning occurs. The set of decision epoch is identical to the set of re-plannings, as each decision epoch implies a re-planning.

In the creation of initial schedules, the schedules can be made *robust* to be able to cope with changes occurring during the day. Robustness is usually incorporated by adding buffers in the schedules to absorb changes. By necessity, buffers imply periods of idle workers, which contradicts the commonly used planning objective of minimizing cost. The inclusion of robustness is then a multi-objective scheduling problem, which aims at finding a trade-off between minimum cost and acceptable robustness.

In both robust and non-robust schedules, failing to react to changes invariably leads to deterioration of the schedule's effectiveness. This means that even for robust plans, re-planning can be beneficial to maintain effectiveness. In the case of severe changes, re-planning is required to ensure that the plan is feasible.

The uncertainty that prompts changes and decision epochs may stem from a variety of sources. In the following, we describe two ways of re-planning, which is separated by the type of uncertainty, and the objectives of the re-planning optimization. Dynamic problems are presented in Section 6.3.1 and disruption management is reviewed in Section 6.3.2.

6.3.1 Dynamic Optimization Problems

Dynamic optimization problems consider reactions to changes that occur during the day of operation. The goal of dynamic optimization is to create updated schedules that incorporate the changes, while minimizing costs (or maximizes profits). A main cause for dynamic properties is when the late arrival of orders means that not all tasks are known at the beginning of the day of operation, and re-planning is then required to accommodate the new tasks into the plans.

This is relevant in settings such as courier mail services (see Larsen et al. [69]) or within health care (see e.g. Cordeau and Larporte [36]), or in settings involving on-call repair or service personnel. Other causes of uncertainty in dynamic problems can include vehicle breakdowns or alterations in travel times [51, 63].

Tasks in dynamic problems can be divided into *advance* tasks, which are known before the day of operation, and *immediate* tasks, which arrive during the day. Depending on the type of problem, different priorities can be given to each type of task. in some cases, advance requests are given higher priority to reward requests that are given early and allows advance planning. The transportation of passengers with reduced mobility presented in Paper D is an example of this approach, by giving preference to passengers who have *prebooked*. In other cases, immediate tasks represent emergencies that must be considered with high priority. Borenstein et al. [17] considers this case for telecommunication technicians.

A proposed measure for dynamic problems is the so-called *degree of dynamism* which measures the ratio of advance and immediate tasks [69]. Variants of this measure can be used to also consider the arrival times and reaction time (the time from the request arrives until its latest scheduling time).

Dynamic problems have been studied for a number of application settings. In the vehicle routing literature, dynamic variants of a number of existing problems have been considered [84] Dynamic systems for route planning are becoming more and more widespread due to advances in information and communication technology according to a recent survey [70]. Other areas considered are workforce scheduling [17] and supply chain management [89].

Solution strategies for dynamic problems may be divided into two groups according to Chen and Xu [29]. *Local* approaches, which considers only orders available at the decision time and solves a static problem on the available orders. *Look-ahead* approaches considers also the expected arrival of additional work which is regarded with some uncertainty.

The local approaches are in two senses the simplest. First, they consider only work currently available at the decision epoch, which makes the problem smaller than the full problem. Secondly, it may be difficult to model certain and uncertain work at the same time. A main complication in look-ahead algorithm is therefore how to integrate probabilistic future work with already received tasks, see e.g. Goto et al. [55].

Commonly, dynamic problems are event-driven where each incident triggers a decision epoch, implying that re-planning occurs at the arrival of each new incident. Dynamic problems can require many decision epochs if the triggering events occur frequently and it seems intuitive to assume that frequent re-

planning provides a more fluent responses to dynamic systems. Therefore, dynamic problems are often solved by fast and simple heuristics, see e.g. Larsen et al. [69] or Regan et al. [86].

Using more complicated methods to solve dynamic problems will naturally require higher solution times. However, as later work is uncertain there is no guarantee that better solutions will come from more time-consuming solution methods. Ghiani et al [52] note that the level of uncertainly in future demand influences the success of more complex solution methods.

In the study of Powell et al.[83] users are allowed to reject changes if they have more accurate information than is present in the model. In the simulations performed in the study, simple greedy heuristics was shown to perform comparably or better than optimal methods.

6.3.2 Disruption Management

Disruption management forms a separate approach to real-time optimization, in which a key component is an existing schedule that reflects "normal" operations. As such, some of the main objectives of disruption management are to minimize the changes made to the schedule and to return to completely normal operations as soon as possible.

Disruption management has been studied extensively for transportation problem that operate on a fixed schedule, particularly airlines where deviances from the flight schedule can be extremely costly. Overviews of airline disruption management are provided in Kohl et al- [67] and Clausen et al. [35].

A related problem of airline crew recovery is studied in e.g. Yu et al. [105] and Nissen and Haase [82]. Disruption management is highly relevant in a number of other areas, including logistics, manufacturing and project scheduling. An overview of applications and solution methods is given by Yu and Qi [106].

A common trait for disruption management is that the decision epochs are not triggered by individual changes. Instead they may be the result of a planners experience that a replanning is needed, or occur continuously at fixed time intervals to ensure that the number of unhandled updates are kept low at all times.

In some applications, schedule changes arrive continuously and a while single update may not be significant to trigger a decision epoch, the many updates will eventually compound to a state that requires replanning. In these cases,

it is impossible to derive decision epochs from the arrival of updates. This is particularly true in airports, where updated information on arrival and departure times, passenger numbers, baggage information and stand allocations are constantly being transmitted. According to Dorndorf, the frequency of airport updates can be as high as several per second [45]. In ground handling, the majority of tasks are linked directly to properties in these airport updates, such as a flight's arrival or departure time, gate assignment, etc. This means that real-time planning for ground handling must include elements of disruption management.

In ground handling problems resembling mobile workforce scheduling, the replanning model can be a combination of dynamic optimization and disruption management. Some information updates are both significant and immediate enough to warrant a direct decision epoch, for example a team finishing a task late or an aircraft being redirected to another stand. Other changes may require less immediate attention, such as updates to a flight's estimated arrival or departure, which is some time away. As these updates happen frequently, it is often not practical to trigger a decision epoch for each individual update. in practical cases, some crew will prefer to have an overview of the next few tasks that do not change repeatedly. Instead, they prefer working from a predominantly fixed schedule, from which changes should be kept minimal. In short, ground handling operations can contain aspects of both dynamic optimization and disruption management.

Chapter 7

Overview of Papers

This chapter introduces the papers of this thesis, which are provided in full length in Part II. A brief introduction to each paper is given, along with details of how the work has been disseminated.

7.1 Paper A: A Dynamic Programming-Based Heuristic for the Shift Design Problem in Airport Ground Handling

This paper presents a construction heuristic for the heterogeneous shift design problem where there can be several demands and each shift can help cover a specified subset of the demands. The heuristic iteratively calculates non-overlapping sequences of shifts which cover a one-dimensional projection of the workload demand. The best sequence of shifts is determined by considering several objectives, including overstaffing, understaffing, skill usage, and cost. The "optimal" sequence of shifts is determined using a dynamic programming recursion. Furthermore, the paper discusses the complexity of shift planning in relation to the flexibility allowed for the shifts and the consequences for the size requirements and running time for the dynamic programming recursion.

The developed algorithm integrates with the WorkBridge *Prepare* software system for workforce planning. The heuristic can be used within the system as a stand-alone heuristic for shift design or as a constructive heuristic for providing an initial solution for the local search algorithm described in Section 7.2.

- The paper is submitted to *European Journal of Operational Research*, 2010.

7.2 Paper B: A Rule-Based Local Search Algorithm for General Shift Design Problems in Airport Ground Handling

This paper introduces a local search metaheuristic for the shift design problem based on simulated annealing. The algorithm uses multiple neighborhoods to efficiently explore different parts of the solution space and an adaptive scoring framework to select neighborhoods based on their past performance. A combination of relaxed constraints and a loosely coupled rule engine provides a tradeoff between constraint satisfiability, flexibility of navigating the solution space, and an highly extendable software base.

The algorithm is evaluated using different heuristics for generating initial solutions and is applied to problem instances from real-life operations in airport ground handling.

- Preliminary work was presented at the ALGO-talk seminar series at DIKU in 2007.

- The work was presented at *3rd Nordic Optimization Symposium* in 2009.

- The algorithm has been implemented in the WorkBridge *Prepare* software system for manpower planning.

- The paper is submitted to *European Journal of Operational Research*, 2010.

7.3 Paper C: Dynamic Routing of Short Transfer Baggage

This paper describes a greedy algorithm for dynamic dispatch of baggage transportation vehicles for delivering bags of transferring passengers in a major European airport. A greedy algorithm is presented that constructs a route by using a score function that considers several measures of desirability of delivering each available bag. The score function uses piece-wise linear time-dependent functions to model complex aspects of the delivery process. Experiments are presented that show that the algorithm performs well for generated scenarios with increasing levels of disruption.

- This work was presented at the ORSEM seminar at DTU Management in 2009.
- The paper has been submitted to *Transportation Research Part E* in 2010.
- A variant of the described algorithm is currently in operation at the company responsible for the handling of all transfer baggage at one of Europe's busiest airports.

7.4 Paper D: Route Planning for Airport Personnel Transporting Passengers with Reduced Mobility

Paper D introduces and solves a model for the scheduling of passengers with reduced mobility (PRM) in a complex airport setting where passengers can be required to travel between terminal buildings and where both arriving and departing aircraft can be parked remotely. To ensure that PRMs are not left unattended, synchronization constraints are imposed on hand-overs of the PRM between agents of different airport areas.

The paper presents a two-level meta-heuristic that separates a declarative decision level with a constraint-observant assignment level. The assignment level is implemented by a greedy insertion heuristic that inserts each PRM into the route network in the order specified by the declarative layer. The declarative order of insertion is modified by Simulated annealing using neighborhood operations. A preference is given to *prebookings*, i.e. advance notices.

The PRM routing problem is a highly complex problem of increasing relevance, since the number of PRMs can be expected to rise with the number of general passengers. Additionally, regulation in the European Union (regulation 1107/2006) has since 2006 mandated that all PRMs in EU airports are entitled to free and fair service. As awareness is increased and the capacity of PRM handling in airport is increased, this is expected to significantly affect the number of PRMs. It is estimated that 10% of the population in EU member states is classified as PRMs [49].

The implemented algorithm is validated on real-life data from a major European airport and show that the algorithm can calculate efficient solutions with a few minutes. This allows the algorithm to be used in a dynamic context, where schedules can be updated to handle delayed flights and the arrival of *immediate* PRMs requiring service.

- A preliminary version of this work was presented at the *2nd Nordic Optimization Symposium* in 2007.
- A variant of the preliminary work was implemented in the WorkBridge resource management system.
- The work in its current form was presented at the ORSEM seminar at DTU Management in 2010 by Line Blander Reinhardt.
- The paper is submitted to *Transportation Research Part B* in 2010.

7.5 Paper E: The Offline Group Seat Reservation Problem

This paper describes a two-dimensional packing problem in which the elements to be packed are fixed in one of the dimensions. The paper is motivated by an application in reserving multiple adjacent seats in a train, where the fixed dimension is the train's route, defining the origin and destination stops of the reservations.

The objective of the problem is to fill the train as efficiently as possible with the available reservations. Two versions are evaluated: In the *knapsack* version of the problem, the train has a fixed size and some reservations may be rejected. In the *bin packing* version, the train consists of a number of smaller cars or compartments and the objective is to minimize the number of cars used to accommodate all reservations.

Although this project was not initially intended for applications in ground handling, some aspects of the problem can be transferred to this application. In areas such as service desk manning the staffing is constant throughout the day. The reservations would then represent tasks with varying capacity requirements to be solved at specific times during the day. The knapsack version determines how many requests can be met for a specific capacity and the bin packing version then determines how many service desks are needed to fulfill all requirements. Of interest for this application is also the bounds considered for the knapsack variant. The bounds $\mathcal{U}_1$ and $\mathcal{U}_2$ split the splits the reservation to multiple single-seat reservations and $\mathcal{U}_4$ splits into single-station reservations. The analogous bounds for the staffing problem split the requests according to capacity requirements or time. Each of the simpler problems produced by the bounds would be of interest to consider in this context as well.

- The paper has been published in the European Journal of Operational Research, 2010.

Chapter 8

Conclusions

In this chapter, the main achievements and contributions of the thesis is reviewed. A summary of the thesis contributions is given below, and a shorter list of major contributions is presented in Section 8.1. In addition, directions for future research within ground handling are discussed in Section 8.2.

The subject of this thesis is the application of operations research techniques to solve both specific and generalized problems in airport ground handling. The venue of ground handling contains a large number of diverse problems, which are subject to the unique environments of modern airports. As many practices are almost identical in any airport, it is possible to provide common models for ground handling problems. Although individual problems have been studied in a number of papers within recent years, the study of generalized models and algorithms for airport ground handling is a relatively new research area.

The work presented in this thesis has been created through cooperation with the industrial partner WorkBridge A/S. During the time spent at WorkBridge, knowledge of ground handling operations has been obtained through cooperation with developers and consultants at WorkBridge, as well as through airport visits and workshops with customer companies.

As a result of the cooperation, the research presented in this thesis is also applied within WorkBridge products. Two of the five presented research papers

describe algorithms which are in operation. In addition, the remaining research conducted as part of this thesis has been disseminated at WorkBridge, thus consolidating the knowledge and experience of operations research techniques within the company.

This thesis is divided into two parts. The first part of this thesis provides a general description of operations research problems arising in the field of airport ground handling. The venue of airport optimization has been introduced, with details of the decision hierarchy of modern airports, as well as an airport's physical layout.

A planning timeline for ground handling problems was presented and problems arising within the model is described. Elements with specific relation to ground handling was considered in detail.

For workforce planning problems, a detailed analysis of demand modeling was presented with emphasis on the airport case, where demand is highly fluctuating and uncertain. Additionally, a review of methods for distribution coverage was presented with extensions to the multi-skill case.

For task-based planning problems, ground handling operations are related to common optimization problems from the literature. The *mobile workforce problem* was identified to cover the majority of ground handling work, and modeling approaches for the mobile workforce problem was discussed.

To the knowledge of the author, Part I of this thesis is among the first to provide a cohesive description of ground handling work with emphasis on operations research.

In the second part of the thesis, five research papers are presented, which cover various projects undertaken within the period of this project. Two papers describe algorithms for the shift design problem, in which shifts must be created to cover a demand, subject to a large number of objectives and constraints.

Paper A presents a construction heuristic based on dynamic programming that iteratively creates shift sequences for the widest line of uncovered demand. The heuristic terminates when the best shift sequence can no longer efficiently cover the remaining demand. Experiments on real-life instances from different ground handling companies show that the heuristic can efficiently cover large shift design problems within a few minutes of running time. In a subsequent study presented in Paper B, the construction heuristic is shown to provide better results than a front-loading heuristic, also presented in Paper B, on almost all instances.

In Paper B an adaptive local search metaheuristic with multiple neighborhoods

based on simulated annealing is presented for the shift design problem. The algorithm allows each neighborhood to satisfy a subset of the constraints, in order to reduce the difficulty of adding new constraints or neighborhoods after its initial deployment. Experimental results are presented on a varying set of real-life instances from ground handling, where the algorithm is consistently shown to significantly improve the results found by the initial runs of a construction heuristic, even within short running times on a normal desktop computer.

An algorithm for dynamic dispatch of vehicles transporting baggage for transferring passengers is presented in Paper C. The algorithm is run when a vehicle returns to the dispatch hall to pick up new bags for delivery, and is used to select the bags to deliver on the vehicle's new route. The algorithm is evaluated using a simulation framework that simulates vehicle behavior under different levels of uncertainty in arrival times of bags and vehicle travel times. Simulation experiments on real-life data show that the case company can achieve a comparable level of service using about half their current fleet size.

A two-stage heuristic for the problem of transporting passengers with reduced mobility is presented in Paper D. The problem is a highly constrained variant of the vehicle routing problem with pickup and delivery, in which multiple agents must cooperate to transport the passengers through different areas of the airport, without leaving passengers unattended. The problem is highly dynamic, where new passengers can arrive continuously throughout the day, and be subject to very short connection times. The algorithm is evaluated on real-life data from a case company, handling averagely 450 passengers a day. the results show that for a short running time of two minutes, a maximum 2% of the passengers was left unscheduled.

Finally, an exact algorithm is presented for the off-line group seat reservation problem, in which reservations for groups of passengers must be assigned to seats within a train to maximize the utilization of the train without separating the passengers of each group. The problem is considered in two variants, a knapsack variant, where the train has a fixed size and the best set of reservations must be selected, and a bin packing variant where all reservations are accepted and the number of train wagons should be minimized. The algorithm considers several relaxations which is applicable in a ground handling context to schedule jobs at service desks. Experimental results are presented, in which it is shown that the proposed algorithm outperforms a standard integer programming model solved using CPLEX.

8.1 Contributions

The main contributions of this thesis consist of the description of the role and application of optimization problems within airport ground handling given in Part I, and in the individual projects presented in the papers of Part II.

- A comprehensive description of planning and scheduling problems in airport ground handling has been provided.
- A discussion of modeling approaches for demand modeling has been provided with specific emphasis on areas which are relevant to airport ground handling.
- A versatile heuristic for a shift design problem with high variance in demand has been developed. The heuristic can consider a large number of constraints and allows a direct trade-off between high running times and the quality of the solutions found.
- A meta-heuristic framework for generalized optimization problems with a varying set of constraints has been proposed. The framework allows several layers of integration of every rule, which allow new rules to be added effortlessly. Good and stable results are shown for real-life data instances of highly varying size and complexity.
- A dynamic algorithm for dispatching baggage for transferring passengers with short connection times has been presented. A weighted selection method using piecewise linear scoring functions is used to model complex constraints within the airport. Simulations on actual data show that the algorithm efficiently dispatches high volumes of baggage with a significantly smaller fleet size than is currently used.
- A simulation framework is presented for the baggage dispatching problem which models several kinds of uncertainty as well as driver reaction to disruptions.
- An algorithm is presented for a complex scheduling and routing of disabled passengers in airports, which successfully combine a high-level decision layer with a greedy heuristic encapsulating complex constraints.

8.2 Directions of Future Research

This thesis presents a number of papers which can each be investigated further within their own scope, as indicated in the papers themselves. There are

many interesting directions of future research for the individual projects, and for ground handling problems in general. We present a few possible directions in this section.

The discussion of demand modeling in Chapter 5 contains several areas to consider. With the current representation of demand, it is difficult to incorporate flexibility (such as task windows) and uncertainty of certain tasks or flights. In the current literature, flexibility is mainly handled by considering the underlying tasks explicitly. However, this may be intractable for large workforces.

Some of the models for the task scheduling problems described in Chapter 6 are well-studied in the literature, such as the vehicle routing problem and the resource constrained project scheduling problem. A less studied problem is the mobile workforce scheduling problem, which is of high relevance for many types of airport ground handling. An interesting direction of future research would be to consider this problem in detail for a broad base of ground handling problems, where the mixture of transportation time and stationary work time varies. A further area of interest for the mobile workforce problem is disruption management, where schedules must be re-planned to handle flight delays.

For the rule-based local search framework considered in Paper B, it would be interesting to consider the framework for other generalized problems, within ground handling or otherwise. It would furthermore be of interest to consider neighborhoods that are more "intelligent" in their selection of new solutions, and how neighborhoods with additional logic perform within the proposed framework.

The two-level simulated annealing heuristic presented in Paper D could be applied to other highly constrained problems. It would in particular be interesting to examine the possibility of using different greedy insertion heuristics. Either by using different heuristics to handle different sets of constraints, or by using different heuristics competitively to modify the solution in different ways.

Part II

Scientific Papers

Paper A

A Dynamic Programming-Based Heuristic for the Shift Design Problem in Airport Ground Handling

Tommy Clausen

Submitted to *European Journal of Operational Research*

A Dynamic Programming-Based Heuristic for the Shift Design Problem in Airport Ground Handling

Tommy Clausen

We consider the heterogeneous shift design problem for a workforce with multiple skills, where work shifts are created to cover a given demand as well as possible while minimizing cost and satisfying a flexible set of constraints.

We focus mainly on applications within airport ground handling where the demand can be highly irregular and specified on time intervals as short as five minutes. Ground handling operations are subject to a high degree of cooperation and specialization that require workers with different qualifications to be planned together. Different labor regulations or organizational rules can apply to different ground handling operations, so the rules and restrictions can be numerous and vary significantly. This is modeled using flexible volume constraints that limit the creation of certain shifts.

We present a fast heuristic for the heterogeneous shift design problem based on dynamic programming that allows flexibility in modeling the workforce. Parameters allow a planner to determine the level of demand coverage that best fulfills the requirements of the organization. Results are presented from several diverse real-life ground handling instances.

A.1 Introduction

The Heterogeneous Shift Design Problem (H-SDP) is concerned with designing a set of work details (shifts) for an organization that specify requirements of different types of employees within the planning period. The shift design problem provide a transformation of the demand from a demand curve (temporal requirements) to more simple shift based requirements. This provides an important tool for organizations to estimate the capabilities of an existing workforce or the need to adjust the workforce to adequately meet the demand.

The shifts should be designed such that they allow the organization to satisfy constraints from labor regulations or the capacity of the workforce. Such constraints include the construction of individual shifts, such as shift lengths, the placement and number of relief breaks or the allowed placement of shifts with regards to office hours or similar requirements. Special volume constraints specify a maximum number of shifts within time periods, which may be used to model the size of the workforce and several regulatory constraints such as limits on nights or weekend shifts.

We consider the shift design problem with specific emphasis on the venue of airport ground handling. There is a large number of tasks that must be carried out for every aircraft that lands at an airport, before it is ready for departure. Many airports are extremely busy and are already operating at full or near-full capacity, whilst expecting further growth in passenger numbers in the future. Furthermore, the emergence of low-cost carriers has created additional demands for short aircraft turnaround times, while the hub-and-spoke network structures of larger carriers are creating demand for reduced connection times between airports. The flight arrival or departure times are often placed at periods of high demands or to provide short waiting times for transferring passengers. This means that airport activity can be extremely high during periods of frequent arrivals and departures, whilst other periods of the day are relatively quiet [9].

The demand at airports is usually represented as a demand curve, in which the planning period is divided into a number of equal-sized periods called *time slots*, each having a number of required workers. Arrival and departure times are planned for five minute intervals, yielding a demand interval length of five minutes as well. The combination of short time slots and flight schedules with high variance in activity means that the demand may contain a high level of *irregularity*, where the difference between requirements of two adjacent time slots can be large.

When the demand is irregular, it is unlikely that a set of shifts can cover the demand perfectly, or even come close. Covering a single peak may require a lot of workers, that will be idle before and after the peak. Therefore, it is up to a planner to find a balance between covering the entire demand and minimizing expenditures. This emphasizes the need to consider both staff shortage (*understaffing*) and surplus (*overstaffing*) as part of the desired solution quality. The desired level of *coverage* for an organization depends on the demand and the available workforce.

The work performed within airports is highly specialized. The ability to perform a specific task may require certain security clearances, training or specialized equipment. The requirements may vary greatly across tasks. Different aircraft types may require different types of equipment; different locations in the air-

port may require different security clearances, and different computer systems (such as check-in terminals) may require different abilities. Many workers may have training in several fields or different certificates, which provide them with capabilities for different partially overlapping areas [10].

The consequence is that shift design for ground handling must handle a heterogeneous demand and workforce. The specialization is modeled by using several demand curves. Similarly, the workforce is divided into groups, each with separate constraints and the ability to cover one or more types of demand.

A.1.1 Workforce Scheduling

The shift design problem may be viewed as part of a more general problem, the *workforce scheduling* or *staff scheduling* problem. In the workforce scheduling problem, timetables (rosters) are created for workers, such that each worker has a sequence of shifts and rest days that in combination meets a specified demand. Various aspects of the workforce scheduling problem consider different levels of detail and relevant data is available or required at different times. It is therefore common to subdivide the problem into several subproblems that can be solved at different times. One such division into five subproblems (or *stages*) is proposed by Tien and Kamiyama [14]: Stage 1 considers the *temporal manpower requirements*, i.e. demand at each time period or shift. In stage 2 the total manpower requirements are determined. Stage 3 considers blocks of recreation days and Stage 4 combines recreation and work days. Finally, Stage 5 assigns shifts to workers. More recently, Ernst et al. [7] provides a comprehensive survey of workforce scheduling. The survey proposes a taxonomy of different modules of which most workforce scheduling references implement one or more.

In the subdivision of Tien and Kamiyama, *shift design* is part of Stage 1, with some overlap into Stage 2. In the taxonomy of Ernst et al. [7], shift design falls under the area of *demand modeling*, in that it transforms *flexible demand* into *shift based demand*.

There are only a few references in the literature that deal directly with shift design. Musliu et al. [12] present a local search algorithm and show experiments for both random and real-life data from a call-centre. Herbers [8] presents an algorithm based on constraint programming for a task-based demand. DiGaspero et al. [5] presents a local search metaheuristic for the *minimum shift design problem*, where the number of different shifts should be minimized. A construction heuristic based on a min cost flow model is also presented.

Dowling [6] describes a metaheuristic based on local search for staff scheduling at an airport. After each neighborhood iteration, the updated solution is checked for feasibility using an external rule engine. Lau [10] considers the *changing shift assignment problem* where the shift scheduling problem is augmented with constraints that specify feasible transitions between shifts. These *shift change constraints* are used to model airport ground handling problems, where required skills are determined by aircraft type.

To the knowledge of the author, no prior references exist in the literature for the heterogeneous shift design problem.

A.1.2 Overview

In this paper we describe a fast construction heuristic for the heterogeneous shift design problem. The heuristic emphasizes the ability to balance demand satisfaction versus minimizing excess shifts, while maintaining an even distribution of coverage shortage and surplus. The heuristic uses dynamic programming to iteratively build sequences of shifts to evenly distribute overstaffing and understaffing across the demand period, while satisfying the constraints of the problem that limits the number of shifts created from different groups.

Experiments are performed on real-life data from ground handling operations in major airports around the world. The data is obtained from the WorkBridge PlanManager software product that specializes in workforce scheduling for airport ground handling. The developed algorithm is intended for integration into WorkBridge PlanManager. The experiments show that good solutions can be found for most instances in less than a minute.

The paper is organized as follows: In Sections A.2 and A.3 we present the H-SDP in detail. Section A.4 presents an overview of the solution strategy for iteratively solving relaxations of the H-SDP. In Section A.5, we define the 0-1 Shift Design problem as a relaxation of the H-SDP that produces a single sequence of shifts and an algorithm for the 0-1 Shift Design problem based on dynamic programming. Performance considerations are discussed in Section A.6. Computational results for diverse problem instances from ground handling are presented in Section A.7 and conclusions are presented in Section A.8.

A.2 The Heterogeneous Shift Design Problem

A main characteristic of the shift design problem is that there are conflicting goals, which can be difficult to obtain simultaneously. In this paper, we consider three conflicting goals: Understaffing and overstaffing, and cost. We shall consider understaffing and overstaffing to be most important, and use cost to describe the preferential placement of shifts in cases where coverage is not affected. If the cost is derived directly from the worker's salary, the algorithm will attempt to avoid expensive shifts, such as nights and weekends. The cost can also be used to model the possibility of adding overtime, the expected availability of temp workers or the preferences of the workers.

When the demand is highly irregular, it is impossible to create solutions which are good in terms of both understaffing and overstaffing. Adding shifts to cover peaks in the demand will introduce a lot of overstaffing as well. Often it will be up to the individual planner to decide the optimal balance between understaffing and overstaffing. A good balance between understaffing and overstaffing will depend on the workforce available and the ability to absorb peaked demand into the shifts. If there are few high, thin peaks, the planner may decide not the cover them, in the expectation that the workers will be able to solve them on the day of operation. In other cases, the planner may require that all demand is covered. The correct balance is obtained by allowing the planner to assign weights to the importance of understaffing, overstaffing and cost.

Sometimes peaks in the demand are explicitly removed by a process known as *peak cutting* before planning is done. However, the weighted approach allows the planner to integrate planning and decision making, rather than making the decision *a priori*.

Another focus of shift planning is the distribution of shifts throughout the planning period. When understaffing and overstaffing exists, it should be distributed as much as possible to increase the robustness of the solution. A good distribution of understaffing increases the chance for the workers to cover more demand than scheduled (and thus further reducing understaffing), as well as the likelihood that enough additional workers from a temp agency will be available. Distributed overstaffing decreases the effect of additional work, illness and other disruptions.

A.3 Basic Notation

Shifts are created for a multi-skilled environment, in which a workload demand may require several skills at once, and a worker may be able to fulfill several types of demand. To model capabilities for workers and demand, we introduce the notion of shift qualifications $q \in Q$ and demand requirements $r \in R$. A requirement (which may include several skills) describes an ability to perform a certain work and a qualification describes all capabilities of a worker following a shift. By using qualifications, shifts are created anonymously, so there is no direct link to the employee that will eventually follow the shift except the implicit expectation that the employee will possess the required skills. In this way, a large degree of flexibility in the workforce is maintained, while the ability to distinguish different employees is preserved.

The shift design problem is specified for at time period, which is subdivided into T smaller time units. Each time unit $t \in [0; T-1]$ has the same duration g (typically 5 minutes) and identifies the time interval $[t \cdot g; (t+1) \cdot g)$. We denote g the *granularity* of the problem. The time period $[0; T)$ is called the planning period and is typically one week or one month.

A.3.1 The Composition of Shifts

A shift specifies the presence of a single worker and the ability to perform certain types of work, thus covering a portion of the demand. A shift s is defined by a qualification q, a start time t_s, and a length l_s. The qualification q serves two purposes. First it determines the expected capabilities of a worker following shift s. Second, it serves as an index for the rules and regulations valid for the worker, so it is assumed that all workers with qualification q work under the same conditions and constraints.

The shifts are typically not allowed to start at every time t, since for $g = 5$ minutes, the resulting set of shifts would be impossible to manage. The allowed starting times are determined by the *shift granularity* g_q. A starting time t is then valid if $t \mod g_q = 0$. The feasible shift lengths are determined by g_q and minimum and maximum lengths $L_q^{\min}$ and $L_q^{\max}$. Typical shift granularities are 15, 30, or 60 minutes, which for $g = 5$ minutes set $g_q = 3, 6, 12$, respectively.

Additionally, a shift may have several breaks in which the worker is not able to cover demand. These breaks are specified by break rules $b(l_s) = (t_{blq}^{\min}, t_{blq}^{\max}, l_{bq})$ that define valid start positions relative to the shift start and the length of the break l_{bq}. The valid start positions of the break are subject to the shift

granularity and the shift length l_s. This allows the break's time window to expand with the shift, and to specify that a break is not relevant for certain shift lengths. A longer shift may for instance require more breaks than a shorter shift. We denote by B_q the set of all break rules for qualification q and by $B = \bigcap_{q \in Q} B_q$ the set of all break rules.

A special kind of break can arise where workers may be allotted time for briefings, wardrobe changes, etc. These are denoted *preparation* and *de-preparation* times and are fixed to the start or end of the shifts. These types of breaks are not uncommon in the airport or ground handling industry [11]. We say in general that a shift is *active* when it is capable of covering demand, and *inactive* otherwise.

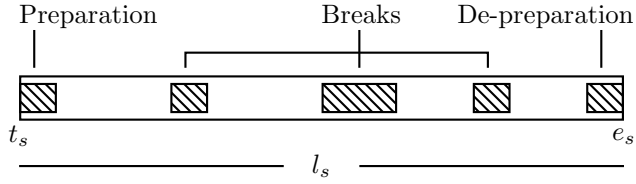


Figure A.1: Illustration of shift composition and related variable names for a shift s. Inactive time from preparation time, de-preparation time and three breaks is shown in hatched boxes.

Depending on the flexibility of the shift parameters, a very large number of different shifts can be created. The number of possible shifts is a key factor in the complexity of shift design and related problems. In particular the added dimensions of breaks increase the number of shifts significantly.

The total number of possible shifts is

$$|\mathcal{S}| = \sum_{q \in Q} \frac{T}{g_q} \cdot \frac{L_q^{\max} - L_q^{\min}}{g_q} \cdot \prod_{b \in B_q} \frac{t_{blq}^{\max} - t_{blq}^{\min}}{g_q} \tag{A.1}$$

We may bound both $L_q^{\max} - L_q^{\min}$ and $t_{blq}^{\max} - t_{blq}^{\min}$ by the maximum shift length, which we may denote $L_{\max}$. This gives the following bound on the number of possible shifts

$$|\mathcal{S}| = O(|Q| \cdot T \cdot L_{\max} \cdot |B|). \tag{A.2}$$

To reduce the size of shifts that needs to be represented in the algorithm, we define a *shift template* s'. A shift template represents all shifts s that have different starting times t_s, but are otherwise identical. This means that any shift s can be obtained from the corresponding shift template s' by adding a

starting time: $s = s'(t_s)$. Each shift template spans l'_s time slots that are either covered or uncovered, so we may consider s' as a binary vector of length l'_s. We denote the set of shift templates S', which by reduction from (A.2) has size

$$|S'| = O(|Q| \cdot L_{\max} \cdot |B|). \tag{A.3}$$

Although the size reduction by T is only linear in the size of the algorithm's input (as D has dimensions $|R| \times T$), the decrease may still be significant. For a planning period of one week with a granularity of 5 minutes, we get $T = 2016$.

A.3.2 Volume Constraints

To model restrictions of the workforce or organizational settings, the overall shift set can be limited by the use of volume constraints. Each volume constraint $v_i = (V_i, v_i^{\max})$ limits the number of shifts allowed within a (not necessarily continuous) time period. The set $V_i \in \mathcal{S}$ contains all shifts affected by the constraint. Thus, we may write each such constraint as

$$\sum_{s \in V_i} s \leq v_i^{\max}.$$

Adjusting the constraint set V_i or the volume limit $v_i^{\max}$ allow volume constraints to model a large number of different scenarios. If $v_i^{\max} = 0$, all shifts in V_i are prohibited, allowing a planner to model closing times or periods where shifts are not allowed to start or end. Setting other values for $v_i^{\max}$ allows the planner to limits varying types of shifts to match the workforce or labor regulations. For example, the total number of shifts, the number of shifts on a single day or the number of certain shift types (e.g. night shifts) may be limited using volume constraints.

The constraint set V_i can easily be adjusted to shift templates by considering the combination of shift template and start time, i.e. $V_i \in \{S' \times T\}$.

$$\sum_{(s',t) \in V_i} s'(t) \leq v_i^{\max}.$$

This adjustment allows us to use shift templates directly in volume constraints.

A.3.3 Workload Demand

The workload demand is represented as a two-dimensional integer matrix $D \in \mathbb{N}_0^{|R| \times T}$. Each entry D_{rt} specifies the required number of active shifts with suitable qualification to cover requirement r at time unit t.

The link between qualifications and requirements can be complex. A shift with qualification $q \in Q$ may be able to cover demand from several different requirements $r \in R$. Similarly, demand for requirement r can be covered by shifts with different qualifications. We say that q covers r, or $q \to r$, if shifts with qualification q can contribute to the coverage of demand of requirement r.

To describe the level of complexity between requirements and qualifications we introduce the *interaction level* of an operation as

$$\text{int} = \sum_{q \in Q} \frac{|\{q \mid q \to r\}|}{|Q|}, \tag{A.4}$$

the average number of requirements covered. Analogously, we may define the interaction of a single qualification as $\text{int}_q = |\{q \mid q \to r\}|$ and for a requirement as $\text{int}_r = |\{r \mid q \to r\}|$.

We assume that requirements are ordered by increasing interaction level, $r_1, \cdots, r_{|R|}$ such that $i < j \Rightarrow \text{int}_{r_i} \leq \text{int}_{r_j}$. We say that r_1 is the *most restrictive* requirement, as it can be covered by the fewest number of qualifications. We similarly assume that qualifications are ordered by decreasing interaction levels, such that q_1 can cover the largest number of requirements.

A shift with qualification q can cover a demand for at most one requirement r at each time t. It is allowed for a shift to cover different requirements at different times. This is because shift qualifications are considered "daily qualifications", which detail the requirement a worker is able to cover within a single day. For practical purposes, this means that given a shift set $\mathcal{S}^*$, the coverage can be calculated independently for each time unit.

A.4 Algorithm Overview

We wish to develop a solution method with fast running times, and which can be terminated prematurely due to time limitations and still produce a solution of reasonable quality. In this way, the method can be used as a stand-alone heuristic and as a construction heuristic for the initial stage of a local search metaheuristic, similar to the approach of DiGaspero et al. [5].

Due to the possible time limitation, we will prefer an algorithm that throughout the majority of its run is able to return a solution that satisfies the volume constraints and has a good coverage distribution across the planning period.

The H-SDP may be formally described as minimizing the weighted sum of understaffing, overstaffing and cost. To achieve the desired distribution of shifts in regard to the demand, understaffing and overstaffing is squared. As we consider cases where coverage takes precedence over cost, the weight of the cost term is set lower than the others.

The integer programming model of the H-SDP is

$$z^* = \min \phi_u \sum_{t=1}^{T} \sum_{r \in R} (u_{rt})^2 + \phi_o \sum_{t=1}^{T} \sum_{q \in Q} (o_{qt})^2 + \phi_c \sum_{i=1}^{n} c_i x_i \tag{A.5}$$

$$\text{s.t.} \sum_{i=1}^{n} a_{it} d_{ir} x_{it} + u_{rt} - \sum_{q \in Q} p_{iq} o_{qt} \geq D_{rt} \qquad \forall r, 1 \leq t \leq T \tag{A.6}$$

$$\sum_{i=1}^{n} a_{it} x_i + \sum_{r \in R} u_{rt} + \sum_{q \in Q} o_{qt} - \sum_{r \in R} D_{rt} = 0 \qquad 1 \leq t \leq T \tag{A.7}$$

$$\sum_{i=1}^{n} v_{ij} x_i \leq v_j^{\max} \qquad 1 \leq j \leq V \tag{A.8}$$

$$x_i, u_{rt}, o_{qt} \in \mathbb{N}_0 \tag{A.9}$$

Here, c_i is the cost of shift i, $1 \leq i \leq n$ and $a_{it} = 1$ if shift i is active at time t, $1 \leq t \leq T$ and $a_{it} = 0$ if shift i is not active at time t. $d_{ir} = 1$ if shift i can cover requirement r and $p_{iq} = 1$ if shift i has qualification q. D_{rt} is the demand for requirement r at time t. The decision variable x_i determines the quantity of each shift type i. Additionally, the decision variable u_{rt} and o_{qt} determines the understaffing of requirement $r \in R$ at time t and the overstaffing of qualification $q \in Q$ at time t, respectively.

Each volume constraint v_j is specified by the constraint limit $v_j^{\max}$ and the incidence matrix V, where $v_{ij} = 1$ if shift type i is affected by rule j, and $v_{ij} = 0$ otherwise.

The objective function (A.5) is a weighted sum of the total shift cost, total understaffing, and total overstaffing using the weight factors ϕ_c, ϕ_u and ϕ_o, respectively. Constraint (A.6) links shifts, understaffing and eligible overstaffing to the demand at each time unit. Constraint (A.7) enforces that the total amount of shifts, understaffing and overstaffing adds up, ensuring that a unit of understaffing or overstaffing is only used once. Constraint (A.8) ensures that the volume constraints are not violated. Finally, constraint (A.9) states that the shift, understaffing, and overstaffing variables are positive integers.

The formulation (A.5)–(A.9) defines an explicit formulation of the SDP, where

any possible shift is represented by a column. This is in a sense an expanded model of the problem, since each shift is directly represented. This means that each shift template s' is represented a large number of times with different starting times.

This kind of model is typically used in the related shift scheduling problem, where it was originally proposed by Dantzig [4] as a set covering model. As the number of shift templates grows rapidly with the flexibility of the shift design, the model can become very large. For this reason, several implicit models have been proposed to reduce the size of the model. Most often the size is reduced by adding special *forward* and *backward* constraints to handle break placements, thus reducing the number of shifts considerably. See e.g. [1], [2], [8], and [13].

In this paper we use a modeling similar to the explicit model (A.5)–(A.9). Instead of reducing the problem size by implicit modeling, we use shift templates instead of shifts to obtain a lesser reduction in problem size. We then solve a relaxed version of the problem.

A.4.1 Iterated Relaxations

The main idea of the algorithm is to perform a two-step relaxation of the H-SDP to get a series of *inner* and *outer* subproblems. The outer subproblem considers one requirement at a time. For each requirement, a number of inner subproblems are solved, each producing a single non-overlapping sequence of shifts. Each outer subproblem terminates when no shifts are returned from the inner subproblem.

The outer subproblem splits the multi-requirement problem into a series of single-requirement problems. That is, we split the H-SDP with demand matrix D into separate subproblems SDP_r with demand vector $d = D_{r*}$. We iteratively solve the subproblems SDP_r, and for each restriction r we consider only qualifications that can cover r.

The relaxed subproblem is then essentially a single-skill shift design problem, although coverage is still evaluated across all subproblems. This means that any understaffing or overstaffing in the solution of one subproblem will be carried to the next. Also, the same qualification can occur in more than one subproblem, so the volume constraints will become more restrictive during the iterations. Each subproblem SDP_r is then still somewhat heterogeneous, although less than the original problem. The amount of information lost in the subdivision depends on the interaction level of the problem instance. To minimize the effect of previously considered subproblems, we consider the requirements in order of

restrictiveness, starting with the most restrictive requirement, r_1.

The inner subproblem subdivides SDP_r even further. Instead of considering demand as an integer vector $d \in \mathbb{N}_0^T$, we consider a binary vector $d' \in \{0,1\}^T$ that for each time t denotes if uncovered workload remains, i.e. if $u_{rt} > 0$. Having reduced the dimension of the demand, we can analogously seek a "one-dimensional" solution to the problem as well. The relaxation of SDP_r is therefore to find a sequence of shifts $\sigma \in \mathcal{S}$ that covers d' as well as possible without violating any of the volume constraints v_i. We denote the binary subproblem 0-1-SDP_r. To avoid cases where 0-1-SDP_r repeatedly returns valid but poor shift sequences, we only return a sequence if it satisfies the termination criteria $\max_u$ and $\max_o$ that limits the maximum understaffing and overstaffing allowed for a sequence.

The conceptual idea of the algorithm is summarized as follows: For each requirement r_i, we solve the subproblem SDP_r by iteratively creating sequences of shifts as specified by the 1-dimensional inner subproblem 0-1-SDP_r. The conceptual algorithm is sketched in Algorithm 1.

Algorithm 1 Pseudocode of conceptual algorithm for solving the H-SDP

```
S* ← ∅
for r ← r_1 to r_|R| do
    Q_r ← {q | q → r}
    d_r ← D_{r,*}
    repeat
        σ* = Solve 0-1 SDP_r(Q_r, d_r)
        S* ← σ
    until σ* = ∅
end for
return
```

A.5 Solving the 0-1 Shift Design Problem

For the 0-1 shift design problem, a number of simplifications can be made which allow the problem to be solved more efficiently than the full H-SDP. As both the demand vector d' and the sequence of created shifts σ is one-dimensional, the volume constraints and coverage measures can be calculated efficiently. The sequence can be considered both as set of shifts: $\sigma = s_1 \cup s_2 \cup \cdots \cup s_n$; and as a set of shift templates: $\sigma = s'_1(t_1) \cup s'_2(t_2) \cup \cdots \cup s'_n(t_n)$. We will mainly use the latter representation in the following.

When considering a non-overlapping sequence of shifts σ, at most one shift can contribute to the demand at any time t. Therefore, we can consider σ as a binary vector $\sigma \in \{0,1\}^T$, which is obtained by concatenating the binary representations of the shift templates in σ.

When considering sequences of shifts for the 0-1-SDP$_r$, a lot of the complexities of the original H-SDP is removed. To compensate for this, the objective function is in some ways simpler, but require other terms to compensate for the simplifications. The quadratic coverage terms of the original problem u^2 and o^2 have been reduced to linear versions $u(\sigma)$ and $o(\sigma)$ since there is no effect of squaring the coverage of a $0-1$ demand. Instead, we rely on the iterated solution method to distribute the coverage. To ensure that the least capable shift is used whenever possible, we minimize the shift sequence's overall interaction level $\text{int}(\sigma)$. This will prioritize the least capable shifts for the currently considered requirement, and save the more capable shifts for less restrictive requirements considered in later iterations, where they are most likely to be usable.

As a technical term, we also introduce the *coverage* of a sequence on a 1-dimensional demand d' as the number of time units where $d'_t = 1$ and $\sigma_t = 1$.

For a shift sequence, the set of volume constraints can be combined into a single binary matrix $V' \in \{\textbf{true}, \textbf{false}\}^{T \times |S'|}$ that for each combination of shift template s' and start time s_t determines if the corresponding shift is legal. We write $V'(\sigma) = \textbf{true}$ if $V'[s_t, s'] = \textbf{true}$ for all shifts in σ. When computing a sequence, V' is considered static, so it is possible to construct a sequence where each individual shift is legal, but the entire sequence violates a volume constraint (since several shifts contribute to the same constraint). There are several ways to handle this, the simplest being to allow small violations to the value constraints. Another simple method is to arbitrarily remove a shift from the sequence, if it contributes to a violated volume constraint. After each sequence is created, V' is updated to reflect the new shifts.

The terms of ϕ and V' can be calculated efficiently by using simple operations.

$$u(\sigma) = \|d'_t \wedge \neg\sigma\|_1 \tag{A.10}$$

$$o(\sigma) = \|\neg d'_t \wedge \sigma\|_1 \tag{A.11}$$

$$cov(\sigma) = \|d'_t \wedge \sigma\|_1. \tag{A.12}$$

$$c(\sigma) = \sum_{s \in \sigma} c(s) \tag{A.13}$$

$$\text{int}(\sigma) = \sum_{s \in \sigma} \text{int}_{q(s)} \tag{A.14}$$

$$V'(\sigma) = \bigwedge_{s \in \sigma} V'(s) \tag{A.15}$$

Here, $\|\cdot\|_1$ denotes the Manhattan norm, which is the sum of the vector elements. For the binary vectors used here, this corresponds to counting the number of ones in the vector. This problem is also known as the *population count* or *popcount* problem and can be solved efficiently by using e.g. the `HAKMEM` method [3].

We use a weighted sum of these measures as the objective function for the 0-1 SDP:

$$\phi(\sigma) = -\phi_{cov}cov(\sigma) + \phi_u u(\sigma) + \phi_o o(\sigma) + \phi_c c(\sigma) + \phi_{int}\text{int}(\sigma).$$

The relaxed subproblem 0-1-SDP$_r$ may be formally described as an IP model by reducing the original model (A.5)–(A.9) in Section A.4. The relaxed model is

$$z'^* = \min \sum_{i=1}^{n} \phi(x_i) \tag{A.16}$$

$$\text{s.t.} \sum_{i=1}^{n} a_{it}x_i + u_t - o_t = d'_t \quad 1 \le t \le T \tag{A.17}$$

$$\sum_{i=1}^{n} a_{it}x_i \le 1 \qquad 1 \le t \le T \tag{A.18}$$

$$\sum_{i=1}^{n} v_{ij}x_i \le v_j^{\max} - v'_j \qquad 1 \le j \le V \tag{A.19}$$

$$x_i, u_{rt}, o_{qt} \in \{0, 1\} \tag{A.20}$$

With a slightly changed notation, $\phi(x_i)$ is the objective $\phi(s)$ for the shift s identified by x_i. The understaffing variables u_t and overstaffing variables o_t have been simplified, since for a one-dimensional demand, there can only be a single item of either understaffing or overstaffing. As in the H-SDP model (A.5)–(A.9), u_t and o_t are connected to the shifts by constraint (A.17). Constraint (A.18) ensures that there is no overlap in the generated shift sequence. Constraint (A.19) enforces the volume constraints, where v'_j is the shift contribution to volume constraint j obtained in previous iterations. In this way, the rules can be evaluated globally, across the different subproblems. Finally, constraint (A.20) states that the decision variables are binary.

A.5.1 The Dynamic Programming Recursion

We use dynamic programming to solve the 0-1-SDP$_r$. The dynamic programming table ν is two-dimensional and contains time as one dimension and the

maximum number of time slots with uncovered demand as the other. A cell $\nu[u,t]$ on the table indexes a sequence ending at t and has maximum understaffing u.

A partial sequence can be any sequence $\sigma_{t,e}$ starting at time t and ending at time e. We may combine partial shift sequences by concatenation to create new partial sequences for which it holds

$$\sigma_{t,e} = \sigma_{t,t'} \oplus \sigma_{t',e} \tag{A.21}$$

$$\phi(\sigma_{t,e}) = \phi(\sigma_{t,t'}) + \phi(\sigma_{t',e}) \tag{A.22}$$

Some notable cases of partial sequences are the full sequence $\sigma = \sigma_{0,T}$ and a single shift template $s'(t) = \sigma_{t,d_{s'}}$.

Intuitively, the content of any cell $\nu[u,t]$ represents the best legal shift sequence σ of shifts that ends at t and leaves no more than u time slots of workload demand uncovered. To determine the best shift sequence for a table position, we use a restricted version of the objective function $\phi(\sigma)$. Since the understaffing $u(\sigma)$ is explicitly considered in the dynamic programming table, $\phi'(\sigma)$ consists of the remaining terms:

$$\phi'(\sigma) = \phi(\sigma) - u(\sigma) = -\phi_{cov}cov(\sigma) + \phi_o o(\sigma) + \phi_c c(\sigma) + \phi_{int}\text{int}(\sigma).$$

We use $\phi'(\sigma)$ as the dominance criterion of the dynamic programming recursion, so sequence σ_1 dominates σ_2 if $\phi'(\sigma_1) < \phi'(\sigma_2)$. For ease of notation, we use $\nu[u,t]$ to denote both the cell at (u,t) and the partial sequence indexed by cell.

We solve the dynamic programming table using a recursion that gradually builds sequences starting at cell $\nu[1,1]$. We set $\phi'(\nu[1,1]) = 0$ and $\phi'(\nu[1,1]) = \infty$ for all other t and u. From each cell $\nu[u,t]$, the partial shift sequence $\nu[u,t]$ is extended with all shift candidates to produce longer sequences. Thus from cell $\nu[u,t]$ we create sequence $\sigma' = \nu[u,t] \oplus s'$ and we set $\nu[u+u_s, t+l_s] = \sigma'$ if $\sigma' \in V'$ and $\phi'(\sigma') < \phi'(\nu[u+u_s, t+d_s])$. If no valid sequence exists for some understaffing/time pair (u,t) then $\phi(\nu[u,t]) = \infty$. Every cell should satisfy the invariant that the sequence indexed by the cell minimizes ϕ' over all partial sequences ending at t with understaffing less than u:

$$\nu[u,t] = \arg\min_{\sigma_{0,t}} \left\{ \phi'(\sigma) \mid u(\sigma) \leq u \cap \sigma \in V \right\}, \tag{A.23}$$

We may formally write the dynamic programming recursion as

$$\nu[u,t] = \min_{s' \in S'} \begin{cases} 0 & t = 0, \\ \phi'(s'(t-l_{s'}) + \nu[u - u(s'(t-l_{s'})), t-l_{s'}] & V'[s',t] = \textbf{true}, \\ & t \geq l_{s'}, \\ & u \geq u(s'(t-l_{s'})) \end{cases} \tag{A.24}$$

From the recursion it can be seen directly that the invariant (A.23) will be satisfied for all cells.

After running the recursion, each cell $\nu[u,T]$ contains a full non-dominated valid shift sequence with understaffing u or less, if such a sequence exists. Each full sequence can then be selected as the solution σ^* to the 0-1-SDP$_r$. The approach we have chosen is to use the full objective function ϕ by taking

$$\sigma^* = \underset{0 \leq u \leq u_{\max}}{\arg\min} \{\phi'(\nu[u,T]) \mid o(\nu[u,T]) \leq o_{\max}\}. \tag{A.25}$$

If $\phi(\sigma^*) = \infty$ or no sequence with $o(\sigma) \leq \max_o$ can be found, we set $\sigma^* = \emptyset$. If this is the case, no shifts are created and the inner loop of the algorithm terminates. No further shifts are created for the current requirement r_i and the algorithm moves on to requirement r_{i+1}.

For the terms of ϕ' that are calculated using bitwise operations, the terms can also be calculated on partial sequences by adding a few extra bitwise shifts and bitmasks. This allows the calculation of ϕ' to be decomposed into separate partial sequences, as in (A.22).

A.5.2 Algorithm Complexity

The algorithm for solving the H-SDP may be described in detailed pseudocode in Algorithm 2. In the pseudo-code, `1d-Demand` produces the binary vector d' of one-dimensional demand. The function `UpdateRules` creates or updates the matrix V' of allowed shift templates based on the volume constraints. Function `CalculateSequence` uses dynamic programming to produce a sequence of shifts σ^*, which are then added to the solution set $\mathcal{S}^*$.

The complexity of the algorithm depends on the complexities of SDP$_r$ and 0-1-SDP$_r$. The inner problem 0-1-SDP$_r$ builds the dynamic programming table ν and runs the recursion. The size of ν is $T \times \|d'\|_1 \cdot u_{\max}$, where $u_{\max} \leq 1$

Algorithm 2 Algorithm pseudo-code

```
S* ← ∅
u_max ← maximum understaffing ratio
o_max ← maximum overstaffing ratio
for r ← r_1 to r_|R| do
  repeat
    d' ← 1d-Demand(r)
    u' ← u_max · ||d'||_1
    o' ← o_max · ||d'||_1
    V' ← UpdateRules()
    σ ← CalculateSequence(d', V')
    S* ← S* ∪ σ
  until σ = ∅ or u_σ > u' or o_σ > o'
end for
return S*
```

and $\|d'\|_1 \leq T$. For each cell in ν, the recursion loops over all shift templates $s' \in S'$. The evaluation of each shift template s' is done by calculating $\phi(s')$.

By using bitwise operations, $\phi(s')$ can be calculated in time $\log_b(|s'|)$, where b is the number of bits in a cpu register word (usually 32 or 64). For a word size of 32 bits, 5 words can represent a shift spanning 160 time units. With a granularity g of 5 minutes, 5 words can then represent a shift longer than 13 hours. We assume that this is always sufficient for representing shifts, i.e. that $\log_b(|s'|) \leq 5$. We consider $\phi(s')$ to be a constant time operation.

The time complexity of 0-1-SDP$_r$ is then

$$O(T^2 \cdot |S'|).$$

The outer subproblem SDP$_r$ calls 0-1-SDP$_r$ until the termination criteria is met. The number of calls depends on the termination criteria and on the amount of demand. In the worst case, the sequence returned by each call will cover a single unit of demand, so the number of iterations may equal the number of units of demand for requirement r, $\|D_{r*}\|_1$.

The total time complexity of solving SDP$_r$ is then

$$O(\|D_{r*}\|_1 \cdot T^2 \cdot |S'|)$$

As we solve SDP$_r$ once for each $r \in R$, the overall time complexity of the algorithm is

$$O(\|D\|_* \cdot T^2 \cdot |S'|) \tag{A.26}$$

where $\|\cdot\|_*$ denotes the sum of all the elements in the matrix (corresponding to the Manhattan norm of the vectorization of the matrix),

$$\|D\|_* = \|vec(D)\|_1 = \sum_{r \in R} \sum_{t=1}^{T} D_{rt}.$$

The algorithm is polynomial in the factors $|R|$ and T, since the demand matrix D of size $|R| \times T$ is provided as input. The algorithm is pseudo-polynomial in the term $\|D\|_*$, as complexity depends directly on the values of the entries in D, which are given as integers in the input.

Note that even in the optimal case where perfect coverage is achieved, the number of iterations will still equal the highest point of demand in D_r, $\max_t\{D_{rt}\}$, which is also pseudo-polynomial in the size of the input.

The number of shift candidates is given by (A.3) in Section A.3.1. By inserting in (A.26), we get

$$O(|R| \cdot \|D_r\|_* \cdot T^2 \cdot |Q| \cdot L_{\max} \cdot |B|).$$

From this we see that the number of shift templates is also pseudo-polynomial in the size of the input, as the complexity depends directly on the maximum shift length $L_{\max}$ which is given as an integer value. Combined with the other factors contributing to the number of shift templates, the complexity relating to $|S'|$ may increase rapidly.

A.6 Performance Considerations

The running time of the algorithm presented in the previous section is mainly dominated by the time complexity of the dynamic programming recursion. In this section we review several approaches to improving performance by reducing the time of the recursion.

A.6.1 Bi-directional Recursion

The size of the dynamic programming table is determined by the maximum understaffing $\max_u$. Even with a restrictive choice of $\max_u$, the table may get large if there are many ones in the demand vector. However, the need for understaffing increases as the time t increases, so if we were to generate a sequence for only half of the planning period, the expected need for understaffing

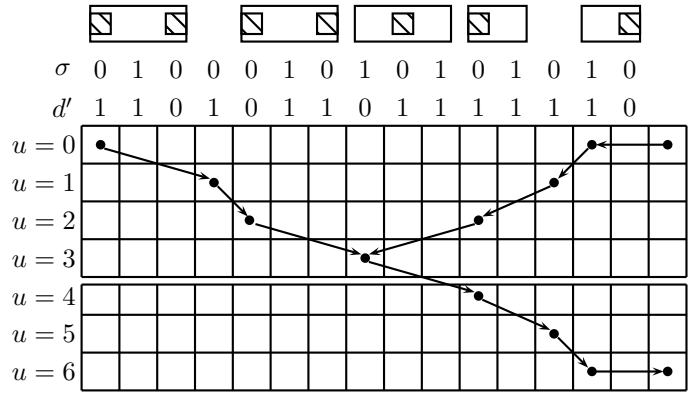


Figure A.2: Illustration of a path through the table for a shift sequence σ. The shifts and associated binary vector are shown on top, above the binary demand vector. The tables shows the space requirements for the forward recursion (center + bottom) and the bi-directional approach (center only).

could be halved as well. This is the idea behind a bi-directional approach, where partial sequences are simultaneously generated from the beginning and end of the planning period and then merged into full sequences. The basic approach and expected reduction in table size is illustrated in Figure A.2.

More formally, the bi-directional approach constructs partial sequences extending *forward* $\sigma^+ = \sigma_{0,t^+}$ and *backwards* $\sigma^- = \sigma_{t^-,T}$. The partial sequences can be combined into a full sequences if $t^- = t^+$, in which case the sequence score is $\phi(\sigma) = \phi(\sigma^+) + \phi(\sigma^-)$. We compute the forward sequences in the interval $[0; T^+]$ and the backward sequences in the interval $[T^-; T]$. To be able to merge the sequences, we must have $T^+ \geq T^-$. The distance between T^+ and T^- must be large enough to allow any full sequence to have at least one shift start/end point within the interval. To achieve this, we set $T^+ = \frac{T}{2} + \frac{L_{\max}}{2}$ and $T^- = \frac{T}{2} - \frac{L_{\max}}{2}$, where $L_{\max}$ is the maximum shift length.

The generation and merging of forward and backward sequences is illustrated in Figure A.3.

When merging forward and backward sequences, all cells in the interval $[T^-; T^+]$ need to be considered. Since the dominance criterion $\phi'(\sigma)$ only considers a single sequence, each cell in $[T^-; T^+]$ can have a value for both the forward and backward recursion. We denote these $\nu^+[u,t]$ and $\nu^-[u,t]$, respectively.

Selecting the best sequence σ^* is similar to (A.25) for the single-directional

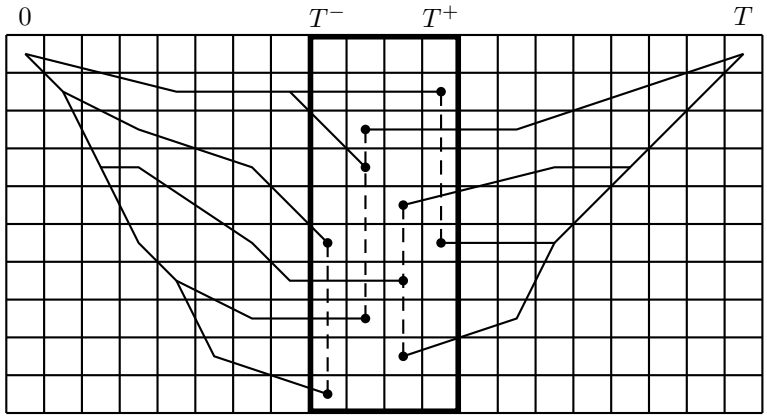


Figure A.3: Merging forward and backward recursion paths. Paths can be merged if they share ending column (dashed).

recursion:

$$\sigma^* = \underset{\substack{0 \le u \le max_u \\ T^- \le t \le T^+}}{\arg\min} \{\phi(\nu^+[u,t]) + \phi(\nu^-[u,t]) \mid o(\nu^+[u,t]) + o(\nu^-[u,t]) \le o_{\max}\}. \tag{A.27}$$

As for the single-directional recursion, the inner loop is terminated if $\phi(\sigma^*) = \infty$ or no sequence is found with $o(\sigma^*) \le o_{\max}$.

A.6.2 Limiting Table Size

In order to limit the running time of the recursion, we limit the state space in several ways, some of which have already been described: The strict dominance of the objectives ensures that the dynamic programming table have only two dimensions, keeping the number of table cells relatively small.

Shift granularities and the volume constraints reduce the number of time slots that can be a start time or end time for a shift. Since the construction of the sequence is only concerned with the transitions between shifts, we remove cells where start or end cannot occur. We denote the set of remaining *active* time units T^*. For instances with restrictive constraints on the placements of shifts due to office hours or low shift granularity, T^* may be significantly smaller than T. The coverage objectives are still calculated on the full demand, taking into account that there may be an uneven distance between two adjacent time units

in T^*.

Another determining factor in the running time of the recursion is the number of shift templates. The set of templates can be reduced by superimposing a more restrictive granularity g^* on the shift granularity g_q. Since g_q is related to both the flexibility of shifts and breaks, the effect of using g^* may be significant. Substituting g^* for g_q also affects the size of T^*.

Finally, the limit on understaffing $u_{\max}$ limits the table size directly, so setting a restrictive limit not only causes the algorithm to terminate sooner, but also reduces the running time of each recursion.

A.7 Computational Results

In this section we review the computational aspects of the dynamic programming algorithm on several instances from real life ground handling operations. We investigate the effect of different combinations of parameters on a subset of the instances, and present full results on all instances on the selected set of parameters. The experiments are focused on balancing overstaffing and understaffing. As a result, we will use a simplified model for cost, and not consider the effects of this parameter in detail.

We introduce the problem instances and their characteristics in Section A.7.1. Parameter effects are presented in Section A.7.2 and finally, large scale results are presented in Section A.7.3.

A.7.1 Problem Instances

The problem instances are taken from real-life instances modeling ground handling operations. There are several different types that varies in the *type* of operation, and in the size and complexity of the problem. The instance type reflects the ground handling tasks performed by the modeled operation. There are several types with different characteristics, as presented in Table A.7.1 Another key characteristic of a problem instance is the *size*, which can be measured along the two axes of the workload demand curve. First, the number of distinct data points T, and secondly the total workload demand, measured as the area of the demand curves, $\|D\|_*$. The final characteristic is the complexity of the operation, which we measure as the interaction level between requirements and

Type	Abbreviation	Description
Passenger Services	`pax`	Terminal work, such as check-in counter staffing and boarding.
Ramp	`ramp`	Aircraft-centric tasks, e.g. pushback or refueling.
Transportation	`trans`	Airport transportation tasks, such as baggage delivery.
Cargo	`cargo`	Loading and handling of cargo.
Operations	`ops`	General operations handling.

Table A.1: Characterization of operation types

qualifications, int, as defined in (A.4). A list of problem instances and their characteristics is provided in Table A.2.

A.7.2 Parameter Tuning Results

In this section we investigate the effects of different parameter values on the solution quality. We group the parameters into several groups, which we address separately in order to limit the overall complexity. To simplify the experiments presented here, we use pre-determined weights for the scoring function $\phi_w = (\phi_o, \phi_{cov}, \phi_u, \phi_{int}, \phi_c) = (100, -10, 1, 0.5, 0.1)$. These weights have been selected to maintain a lexicographic ordering between the individual terms. Overstaffing will always be weighted highest, followed by coverage, and so on. Setting the weights to these values corresponds to a planner having high priority on obtaining a good coverage and lowest priority on minimizing cost.

We review the performance of the algorithm with different sets of parameters using two sample scenarios: `pax.r.a` and `ramp.g.a`. These scenarios have been chosen because they represent each of the two major operation types, `pax` and `ramp` and are not too similar in terms of interaction level and size.

Termination Criteria

We first address the termination criteria $\max_u$ and $\max_o$. Table A.3 presents the effects of running with different termination limits on `pax.r.a` and `ramp.g.a`.

Figure A.4 graphically shows the connection between relative understaffing $u/|D|$, relative overstaffing $o/|D|$ and the running time of the experiments.

Scenario	$\|R\|$	$\|Q\|$	int	T	$\|\|D\|\|_*$
`cargo.m.a`	8	5	0.44	2016	15294
`ops.e.a`	3	5	0.5	8064	17887
`pax.e.a`	6	10	0.21	8064	59329
`pax.g.a`	1	2	0.5	1008	11981
`pax.g.b`	1	2	0.5	1008	14143
`pax.m.a`	10	9	0.52	2016	12796
`pax.r.a`	3	8	0.15	336	3721
`pax.r.b`	3	12	0.15	336	3721
`ramp.e.a`	9	15	0.08	1344	6677
`ramp.e.b`	7	10	0.58	8064	59383
`ramp.e.c`	7	10	0.58	8064	59383
`ramp.g.a`	1	2	0.5	1008	7275
`ramp.g.b`	1	2	0.5	1008	8710
`ramp.m.a`	13	9	0.19	2016	3448
`ramp.r.a`	1	4	0.25	336	1943
`trans.o.a`	6	12	0.5	1008	12925
`trans.o.b`	6	12	0.5	1008	19603

Table A.2: Characteristics of problem instances

As seen in the figure, the selected termination criteria each produce a non-dominated solution, where u cannot be decreased without increasing o. As $u/|D|$ approaches 0, $o/|D|$ becomes increasingly higher (and analogously for $o/|D|$ and $u/|D|$), which means that in many cases a balanced approach is preferable.

All non-dominated sets of parameters may be considered good, so choosing the best combination depends on the priorities of the planner.

The actual levels of coverage obtained by setting different termination criteria are illustrated again in Figure A.5. The figure shows the coverage as contour on the workload for scenario `ramp.g.a` with the two termination criteria $(\max_u, \max_o) = (0.7, 0.4)$ and $(\max_u, \max_o) = (0.8, 0.6)$. The result for $(\max_u, \max_o) = (0.8, 0.6)$ covers more of the demand, but with a higher degree of overstaffing.

From the illustrations we identify $(\max_u, \max_o) = (0.7, 0.4)$ as the most promising combination for obtaining a balance between understaffing and overstaffing, while keeping the running time low. We choose these parameters for further study, as we wish to emphasize this aspect of the algorithm.

Scenario	$u_{\max}$	$o_{\max}$	Iters	o	u	$o/\|\|D\|\|_*$	$u/\|\|D\|\|_*$	$\|\mathcal{S}^*\|$	Time (s)
pax.r.a	0.5	0.2	29	288	4074	0.01	0.18	398	10.25
pax.r.a	0.5	0.4	33	666	3150	0.03	0.14	438	10.48
pax.r.a	0.5	0.6	37	1272	2478	0.06	0.11	465	10.96
pax.r.a	0.7	0.2	38	384	3570	0.02	0.16	448	14.66
pax.r.a	0.7	0.4	57	2472	1596	0.11	0.07	502	17.78
pax.r.a	0.7	0.6	84	4020	504	0.18	0.02	595	20.35
pax.r.a	0.8	0.2	49	576	3294	0.03	0.15	477	18.18
pax.r.a	0.8	0.4	119	4176	456	0.19	0.02	618	28.09
pax.r.a	0.8	0.6	143	4770	210	0.21	0.01	682	29.78
pax.r.a	0.9	0.2	179	3708	846	0.17	0.04	589	43.92
pax.r.a	0.9	0.4	287	5166	114	0.23	0.01	722	55.89
pax.r.a	0.9	0.6	290	5214	78	0.23	0	725	54.61
ramp.g.a	0.5	0.2	16	1036	4450	0.07	0.31	151	2.35
ramp.g.a	0.5	0.4	20	1926	2904	0.13	0.2	185	2.65
ramp.g.a	0.5	0.6	24	3120	1686	0.21	0.12	218	3.03
ramp.g.a	0.7	0.2	28	1590	3360	0.11	0.23	176	4.71
ramp.g.a	0.7	0.4	39	3506	1436	0.24	0.1	230	5.64
ramp.g.a	0.7	0.6	44	4454	932	0.31	0.06	250	6.47
ramp.g.a	0.8	0.2	44	2260	2554	0.16	0.18	197	7.91
ramp.g.a	0.8	0.4	61	4292	1010	0.29	0.07	248	9.22
ramp.g.a	0.8	0.6	68	5178	648	0.36	0.04	267	9.77
ramp.g.a	0.9	0.2	96	3812	1454	0.26	0.1	251	14.38
ramp.g.a	0.9	0.4	151	6694	412	0.46	0.03	311	17.06
ramp.g.a	0.9	0.6	170	7718	296	0.53	0.02	330	17.9

Table A.3: Effects of altering termination parameters for the recursion. $\max_u$ and $\max_o$ are the termination limits, o and u are the resulting units of over- and understaffing and $o/|D|$ and $u/|D|$ presents the coverage ratio to the total demand. $|\mathcal{S}^*|$ is the number of created shifts.

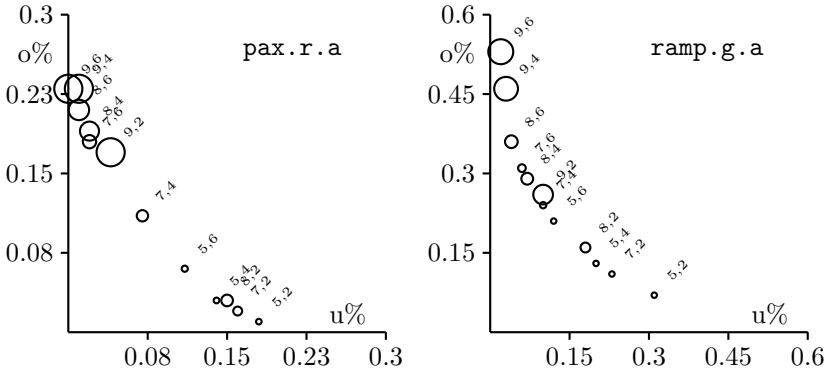


Figure A.4: Illustration of results for different termination criteria for scenarios `pax.r.a` (left) and `ramp.g.a` (right). Each circle represents the results of a run. The position indicates the resulting mix of relative understaffing and overstaffing. The diameter of the circle represents the runtime. Used termination criteria $u_{\max}, o_{\max}$ ($\times 10$) is shown next to each run.

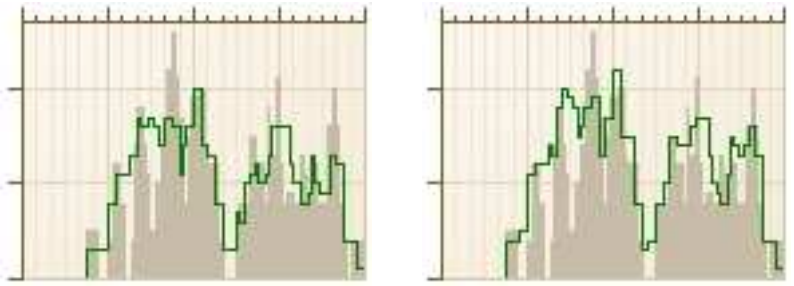


Figure A.5: Coverage for a sample day using termination criteria $(\max_u, \max_o) = (0.7, 0.4)$ (left) and $(\max_u, \max_o) = (0.8, 0.6)$ (right) for scenario `ramp.g.a`.

Scenario	g^*	B-D	Iters	Cells	$\lvert S' \rvert$	$o/\lVert D \rVert_*$	$u/\lVert D \rVert_*$	$\lvert \mathcal{S}^* \rvert$	Time (s)
`pax.r.a`	60	N	57	50913.49	239.21	0.11	0.07	502	18.16
`pax.r.a`	60	Y	52	26357.06	242.46	0.07	0.11	481	10.58
`pax.r.a`	30	N	54	82778.50	1284.33	0.02	0.02	492	148.16
`pax.r.a`	30	Y	54	41149.57	1298.59	0.02	0.02	481	85.86
`ramp.g.a`	60	N	39	60447.15	31	0.24	0.1	230	5.83
`ramp.g.a`	60	Y	40	29840.85	31	0.24	0.11	232	4.46
`ramp.g.a`	30	N	37	124157.95	91	0.22	0.09	225	28.05
`ramp.g.a`	30	Y	39	61801.72	91	0.22	0.09	231	18.03

Table A.4: Effects of altering shift granularity and recursion direction for the sample scenarios. B-D indicates whether the bi-directional recursion is used. Cells and $|S^*|$ is the average number of visited cells and shift templates per iteration. $o/|D|$ and $u/|D|$ is the relative amount of over- and understaffing compared to the total workload demand. $|\mathcal{S}^*|$ is the resulting number of shifts.

Performance Parameters

In this section, we review the effect of two parameters that directly influences the size of the problem: The bi-directional recursion, and the granularity of the used shift templates. The finer the granularity, the more different shift templates are allowed. We run experiments with a modest setting of 60 minutes and a more fine-grained setting of 30 minutes. Results of the experiments are presented in Table A.4.

As seen, the bi-directional recursion is consistently faster than the single-direction version, as the average number of cells visited per iteration can be roughly halved. The time savings are in the range 25–50%, depending on the size of the problem.

Increasing the granularity greatly increases the number of shift templates S' and consequently, the running time is also increased. As an example, consider the single- and bi-directional run on pax.r.a. The exact same number of shifts was created in both cases, but with significantly better results using the 30 minute granularity. However, the running time was also increased significantly from 10 to 90 seconds.

We observe from the results that the bi-directional recursion provides a significant improvement in running times. We identify the bi-directional recursion with 60 minute granularity as the most promising candidate for the declared goal of quickly creating solutions of reasonable quality. If the focus instead was on higher solution quality, the 30 minute granularity is clearly preferred.

Scenario	Iters	$\lvert T^* \rvert$	$\lvert S' \rvert$	$o/\lVert D \rVert_*$	$u/\lVert D \rVert_*$	z^*	$\lvert \mathcal{S}^* \rvert$	Time (s)
cargo.m.a	28	59	7.86	0.03	0.42	1690.32	90	1.17
ops.e.a	11	587	38	0.11	0.24	2806.56	208	27.95
pax.e.a	34	587	84	0.1	0.13	11011.25	791	178.82
pax.g.a	64	560	31	0.23	0.14	51827.83	399	33.15
pax.g.b	74	560	31	0.23	0.13	67266.31	469	38.48
pax.m.a	37	59	6.68	0.17	0.56	8513.59	88	1.87
pax.r.a	52	153	242.46	0.07	0.11	5666.14	481	10.2
pax.r.b	63	153	145.92	0.12	0.06	2414.79	521	5.98
ramp.e.a	35	504	66.46	0.07	0.48	3442.19	606	14.03
ramp.e.b	45	517.8	30.47	0.22	0.26	34674.76	767	85.52
ramp.e.c	45	227.33	16.64	0.2	0.22	43681.81	780	26.71
ramp.g.a	40	140	31	0.24	0.11	23628.24	232	4.32
ramp.g.b	47	140	31	0.23	0.1	31522.56	278	5.22
ramp.m.a	17	59	13.88	0.06	0.67	607.73	15	1.63
ramp.r.a	23	153	111	0.09	0.1	974.59	276	2.29
trans.o.a	110	147	22	0.29	0.2	50372.61	462	10.05
trans.o.b	111	224	22	0.26	0.19	57942.08	664	25.01

Table A.5: Results of running with the determined parameters on all scenarios. Iters shows the number of iterations. $\lvert T^* \rvert$ and $\lvert S' \rvert$ shows the average value of T^* and the average number of shift templates for the iterations. $o/\lVert D \rVert_*$ and $u/\lVert D \rVert_*$ shiws the relative staffing levels. z^* shows the total objective function value and $\lvert \mathcal{S}^* \rvert$ shows the number of created shifts.

A.7.3 Dynamic Programming Results

This section presents the results for the bi-directional dynamic programming heuristic with 60 minute granularity. Results of running the heuristic on all scenarios are presented in Table A.5. Table A.5 shows that 10 of the 17 instances are solved within 15 seconds and 15 are solved within one minute, with the final two instances solved in 94 and 191 seconds. Table A.5 also shows a large differences in the size reductions possible for the instances. The largest scenario, pax.e.a has an average of 587 active time units (out of 8064 total) and 84 shift candidates. In comparison, the scenario `ramp.e.c` is of comparable size, but was reduced much further to 227 active time units and 16 shift candidates. This instance was solved in 26 seconds, which illustrates the effectiveness of the size reductions.

A.8 Conclusions

We have presented an algorithm for the heterogeneous shift design problem that emphasizes an even distribution of shifts throughout the planning period.

Furthermore, the algorithm is highly customizable in the amount of working rules satisfied and the trade-off available between running time and solution quality.

Experimental results are presented for a number of real-life problem instances taken from a variety of airport ground handling operations. The experiments show that the algorithm is able to generate good quality solutions for all problem instances quickly, and that the algorithm provides efficient parameters for balancing understaffing and overstaffing.

References

[1] I. Addou and F. Soumis. Bechtold-jacobs generalized model for shift scheduling with extraordinary overlap. *Annals of Operations Research*, 155(1):177–205, 2007.

[2] S. E. Bechtold and L. W. Jacobs. Implicit modeling of flexible break assignments in optimal shift scheduling. *Management Science*, 36(11):1339–1351, 1990.

[3] M. Beeler, R. W. Gosper, and R. Schroeppel. Hakmem. Memo 239, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Mass., 1972. Item 169.

[4] G. B. Dantzig. A comment on edie's "traffic delays at toll booths". *Journal of the Operations Research Society of America*, 2(3):339–341, 1954.

[5] L. Di Gaspero, J. Gärtner, G. Kortsarz, N. Musliu, A. Schaerf, and W. Slany. The minimum shift design problem. *Annals of Operations Research*, 155(1):79–105, 2007.

[6] D. Dowling, M. Krishnamoorthy, H. Mackenzie, and D. Sier. Staff rostering at a large international airport. *Annals of Operations Research*, 72(0):125–147, 1997.

[7] A. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3–27, 2004.

[8] J. Herbers. *Models and Algorithms for Ground Staff Scheduling On Airports*. Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, Faculty of Mathematics, Computer Science and Natural Sciences, 2005.

[9] N. Kohl, A. Larsen, J. Larsen, A. Ross, and S. Tiourine. Airline disruption management-perspectives, experiences and outlook. *Journal of Air Transport Management*, 13(3):149–162, 2007.

[10] H. C. Lau. On the complexity of manpower shift scheduling. *Computers & Operations Research*, 23(1):93–102, 1996.

[11] A. J. Mason, D. M. Ryan, and D. M. Panton. Integrated simulation, heuristic and optimisation approaches to staff scheduling. *Operations Research*, 46(2):161–175, 1998.

[12] N. Musliu, A. Schaerf, and W. Slany. Local search for shift design. *European Journal of Operational Research*, 153(1):51–64, 2004.

[13] M. Rekik, J.-F. Cordeau, and F. Soumis. Implicit shift scheduling with multiple breaks and work stretch duration restrictions. *Journal of Scheduling*, 13(1):49–75, 2009.

[14] J. M. Tien and A. Kamiyama. On manpower scheduling algorithms. *SIAM Review*, 24(3):275–287, 1982.

Paper B

A Rule-Based Local Search Algorithm for General Shift Design Problems in Airport Ground Handling

Tommy Clausen

Submitted to *European Journal of Operational Research*

A Rule-Based Local Search Algorithm for General Shift Design Problems in Airport Ground Handling

Tommy Clausen

We consider a generalized version of the shift design problem where shifts are created to cover a multiskilled demand and fit the parameters of the workforce. We present a collection of constraints and objectives for the generalized shift design problem. A local search solution framework with multiple neighborhoods and a loosely coupled rule engine based on simulated annealing is presented. Computational experiments on real-life data from various airport ground handling organization show the performance and flexibility of the proposed algorithm.

B.1 Introduction

The shift design problem (SDP) is an important problem arising in workforce scheduling, which is concerned with assigning the workforce to cover a specified demand. The goal of shift design is to create a set of shifts that cover the demand as well as possible, while satisfying the large number of constraints arising from labor regulations, the composition of the workforce and employee preferences.

The shifts generated from a solution to the shift design problem form the input for subsequent planning stages in workforce planning, such as rostering and crew scheduling. A secondary application for shift design is to determine the size of the workforce required to solve the demand.

It is common to divide workforce scheduling into stages, which are solved sequentially. Tien and Kamiyama [22] present a five stage model for workforce scheduling problems: Temporal (i) and total (ii) manpower requirements; the determination of recreation blocks (iii) and a recreation/work schedule (iv); and

shift scheduling (v). In this setting, shift design may be applied in stage (i) and (ii).

The majority of references for workforce scheduling focus mainly on rostering or crew scheduling (stages (iii)-(iv)). It is common in these problems to assume that the shifts to schedule are either fixed or part of a (limited) candidate set. See e.g. the survey for workforce scheduling and rostering problems by Ernst et al. [9].

In this paper, we consider the shift design problem for the specific venue of airport ground handling, where the majority of the demand is linked to the arrival and departures of flights. The uneven distribution of flights during the day can cause large fluctuations in demand from one time unit to the next. For applications in airports or other places with a highly variable demand, an amount of idle time is unavoidable for workers during the working days. This means that combinatorial algorithms are either very complex or will provide poor bounds on the workforce size actually required. In these cases, a dedicated shift design algorithm is more suited to determine a realistic workforce size or distribution.

Compared to the well-studied workforce scheduling and rostering problems, the shift design problem has by comparison only received limited attention in the literature. Herbers [10] describes a shift design algorithm for operational planning, where demand is given as actual tasks. Musliu et al. [17] present a tabu search algorithm for a general shift design problem. Di Gaspero et al. [6] present a local search algorithm for a variant of the shift design problem, in which the number of different shifts generated is minimized.

Some previous work present systems which consider the generation of shifts as a sub-problem: Dowling at al. [8] present a staff scheduling system for airport operations with flexibility in shift lengths and break placements in which the core algorithm uses a simulated annealing algorithm that validates modified solutions against an external rule engine. Chu [4] present a three step crew planning system for airport baggage services which include duty (shift) generation, crew scheduling and crew rostering. A goal programming-based heuristic is presented for the duty generation problem with fixed length shift with a flexible break. Brusco et al. [2] present a scheduling implementation for airport ground crew in which a column generation based heuristic is used to generate shifts.

Airport ground handling work requires a high degree of coordination between workers with different skills. Lau [14] notes that different aircraft types and airlines may require different skills, possibly yielding a high number of work types, see also Ho and Leung [11]. The ability to perform different skills may be represented by a skill matrix, or hierarchically determined as described by

Hung [12] such that more experienced workers can perform an increasing set of skills.

Several papers (e.g. Bechtold and Brosco [1], Brusco et al. [2]) note the need to plan workers with different contracts, such as fulltime and parttime workers together. In some cases labor regulations require that a specified mix of different contracts are present.

In this paper we present a local search framework for a the shift design problem which can model a variety of real-life shift design problems arising in airport ground handling. The framework is designed for rapid extension to easily include new rules or neighborhoods, to solve new problems or increase performance for the occurrence (or absence) of a specific combination of rules. The presented framework is not intended to solve a specific shift planning problem, but to model very diverse shift planning problems. This enables modern service companies, which have many operations with different workforce characteristics and labor conditions, to centralize and effectivize their planning processes. The need for flexible, general workforce scheduling approaches is noted as a major future challenge by Ernst et al. [9].

The presented algorithm was developed as part of the WorkBridge *Prepare* software product for workforce planning and rostering. The system is targeted service companies with emphasis on airport ground handling. As a component of a general software tool, the developed algorithm is required to handle many different types of operations. To illustrate the flexibility obtained by the algorithm, we present good results for a set of real-life problems instances spanning several companies, types of ground handling and countries of operation.

The remainder of the paper is structured as follows. Section B.2 presents the shift design problem in detail and introduces notation and variables. Section B.3 present the components of the framework, including constraints and objectives. A local search algorithm with multiple neighborhoods and loosely coupled rule engine is presented in Section B.4. Computational results on real-life instances are presented in Section B.5 and conclusions are presented in Section B.6.

B.2 Definitions and Terminology

The shift design problem covers a planning period, which is subdivided into T smaller, equally sized time slots $t = 0, \ldots, T-1$. It is common to use a planning period of one week or one month.

Shifts are to be created for a multi-skilled environment, in which a workload demand may require several skills at once, and a worker may be able to fulfill several types of demand. To model capabilities for workers and demand, we introduce the notion of shift qualifications $q \in Q$ and demand requirements $r \in R$. A requirement (which may include several skills) describes an ability to perform a certain work and a qualification describes all capabilities of a worker following a shift. By using qualifications, shifts are created anonymously, so there is no direct link to the employee that will eventually follow the shift except the implicit expectation that the employee will possess the required skills. In this way, a large degree of flexibility in the workforce is maintained, while the ability to distinguish different employees is preserved.

The workload demand is represented as a two-dimensional integer matrix $D \in \mathbb{N}_0^{|R| \times T}$. Each entry D_{rt} specifies the required number of active shifts with suitable qualification to cover requirement r at time unit t. A simple indication of the *size* of a problem instance is the total amount of demand to cover, which we denote $\|D\|_*$, the sum of all entries in D.

A shift (or duty) defines a continuous work period of a single worker, possibly interrupted by meal or relief breaks. A shift s is defined by a qualification q_s, a start time t_s, and a length l_s. For ease of notation, we also use $e_s = t_s + l_s$ as the end time of shift s. The qualification q_s serves two purposes: First it determines the expected capabilities of a worker following shift s. Second, it serves as an index for the rules and regulations valid for the worker, so it is assumed that all workers with qualification q work under the same conditions and constraints.

Additionally, a shift may have several meal and relief breaks during the shift, during which demand is not covered. Breaks are usually flexible, allowing the break to be placed within a time window. For shifts with several breaks, high flexibility may have a high impact on both the ability to cover demand and the complexity of the problem. Although the position of breaks is rarely determined at this stage of planning, it is still necessary to consider breaks to get an accurate estimation of the demand coverage.

A special kind of idle periods can arise where workers may be allotted time for briefings, wardrobe changes, etc. We denote these *preparation* and *de-preparation* times and are fixed to the start or end of the shifts. These types of breaks are not uncommon in the airport or ground handling industry [2], [15].

B.2.1 Workload Coverage

It is desirable to fit the shifts to the demand as good as possible, thereby minimizing the amount of overstaffing (surplus of shifts) and understaffing (shortage of shifts). It is often unavoidable to have either overstaffing or understaffing in the final solution. When there is a high degree of variance in the workload, it may even be desirable to find a "middle-ground" solution that have both overstaffing and understaffing. In these cases the coverage should be "smoothened" to distribute the overstaffing and understaffing across the planning period. A smooth coverage will in many cases have a higher degree of robustness, as it locally minimizes both coverage problems and excess capacity. There are several ways to model such a smoothing. Chu [4] minimizes the maximum overstaffing occurring across all time slots. Dowling [8] minimizes the sum of the squared deviance from the target. We choose an approach similar to Dowling, where we square each overstaffing per time slot separately for each shift qualification. Similarly, understaffing is squared for each requirement and time slot separately.

To determine the overstaffing and understaffing at each time slot, an assignment problem needs to be solved that assigns (shifts with) qualifications to requirements. Although this is not a difficult problem, it must be solved many times during the course of the algorithm so we solve it by using a greedy heuristic. The heuristic assumes that both requirements and qualifications can be ordered $r_1, \ldots, r_n$ and $q_1, \ldots, q_n$ such that it is always preferable to cover the lowest numbered requirement using the lowest numbered qualification, whenever possible.

The heuristic works as follows: For each requirement starting with r_1, we attempt first to assign shifts with q_1, then shifts with q_2 and so on, until no demand for r_1 remains. The heuristic then continues for r_2 to r_n.

The coverage calculation clearly produces sub-optimal solutions in terms of the total squared coverage, since it gathers overstaffing and understaffing in few groups, rather than distributing them as evenly as possible. However, the heuristic has the advantage that it will assign understaffing to the highest numbered requirements (deemed the least important) and assign overstaffing to the highest numbered qualifications. In many cases, the considered workforce is fully or partially hierarchical, such that higher numbered qualifications are more capable. In these cases, grouping overstaffing on the most capable parts of the workforce provides a more robust coverage.

B.3 Modular Components

To maintain a high level of flexibility for the planner using the system, the constraints and objectives are included as modular components. This means that any constraint and objective can be included or excluded by the planner to achieve the desired model of the workforce and applicable labor requirements.

A key concern when designing a local search heuristic for a generalized set of work rules and objectives is how these items are modeled and evaluated. We have chosen a software development-oriented approach to flexibility in which each rule is represented by a separate modular entity which interfaces both the algorithm and the planner's application. In the planner's application, the rule provides an easily understandable graphical representation that allows the planner to model the workforce with little or no previous training.

In the algorithm, all rules are known through general interfaces by the rule engine, which asserts feasibility, and the objective function for calculating the objective value.

The majority of relevant literature consider a set of rules that is defined a priori and modeled directly in the MIP-model (for MIP-based approaches), or implemented directly in the neighborhood (for local search approaches). See Thompson [21], Brusco et al. [2] and Musliu et al. [17] for examples of this approach.

More flexible approaches to generic rule handling integrate local search and constraint programming, see e.g. Shaw [20] or the general framework of van Hentenryck and Michel [23]. In a similar approach Quimper and Rousseau [19] consider a large neighborhood search algorithm where constraints are described in regular languages and evaluated using automata.

Recent approaches to adding generic rules in MIP models define a domain specific language for defining the rules and then compiling the rule definition into MIP constraints. This approach is used for airline rostering by Kohl and Karisch [13] and nurse rostering by Dohn et al. [7].

The flexibility of the developed algorithm can be viewed as somewhere between the fully flexibly language-based approaches and the fixed a-priori approaches. Although new rules must be implemented in software, the modular framework allows new rules to be deployed as needed, independently of the surrounding system. The addition of new rules can be viewed as a task for a trained consultant rather than a complete software upgrade.

We present a subset of objectives in Section B.3.1 and rules in Section B.3.2 and Additional details and a general representation of rules and objectives are presented in Section B.3.3.

B.3.1 Objectives

The shift design problem is a compromise between many different (and often conflicting) goals. To model this, the objective function $f(x)$ consists of a number of individual objective terms, combined in a weighted sum

$$f(x) = \sum_{o \in \mathcal{O}} \alpha_o \cdot o(x)$$

where $f(x)$ is to be minimized. Each objective o has an individual weight α_o that can be modified by a planner. We consider three main objectives and a number of minor objectives. The three main objectives are overstaffing, understaffing and cost, and are described in detail below.

Overstaffing and understaffing minimizes the surplus and shortage of shifts to cover the demand. Both objectives determine the fit of shifts to the demand, which we with a common term denote *workload coverage*. The coverage terms are the most complex terms of the algorithm. They are described in detail in Section B.2.1.

The cost objective minimizes the expenditure of working the shifts in the solution. It may correspond directly to the salary of the workers or reflect more abstract costs, such as the availability of agency workers, etc. Cost is defined as a base cost with a flexible number of supplements that increase the cost of shifts at specified time intervals, such as nights and weekends.

Setting different weights for the cost, overstaffing and understaffing objectives provide flexibility in modeling a variety of coverage scenarios. For example, it is common in shift scheduling models to require that all demand is covered. This can be achieved by setting a high weight on understaffing, a lower weight on cost and removing overstaffing.

Additionally, a large number of objectives may stem from preferences and penalty terms for the violation of relaxed rules. These are described in detail in Section B.3.2.

B.3.2 Rules

Rules are constraints a planner may add to the problem in order to specify certain traits in the solution found by the algorithm. Rules are flexible and can be modified by parameters. Some rules can be added a number of times with different parameters and most rules can be left out of the algorithm if not needed. This allows the algorithm to model a large variety of different shift design problems.

The composition and placement of shifts may depend on a large number of different rules, which may vary from operation to operation. A selection of rules is presented below:

1. Valid shift start time granularity (e.g. half-hourly).
2. Shift length.
3. Preparation and de-preparation times.
4. Number and placement of breaks. Each break has a duration and associated break window that depends on the length of the shift.
5. Time intervals where no part of a shift can occur.
6. Time periods where shifts cannot start.
7. Time periods where shifts cannot end.
8. The allowed number of shifts.
9. Allowed number of working hours.
10. Allowed relative number of shifts (e.g. an allowed ratio of fulltimers and part-timers).
11. Average shift length.

Rules 1-4 control the composition of a single shift. These rules are considered basic and are not allowed to be violated or considered as soft constraints (although it is possible to specify that shifts should contain no breaks). Rules 5-7 adds additional constraints to the placement of shifts, and can be evaluated for each shift individually. The remaining rules require the state of all shifts covered by the rule to be calculated.

Rules 1 and 3–7 are *declarative*, and specify a fixed set of values that variables must (or cannot) take. The remaining rules are *value*-based and define the

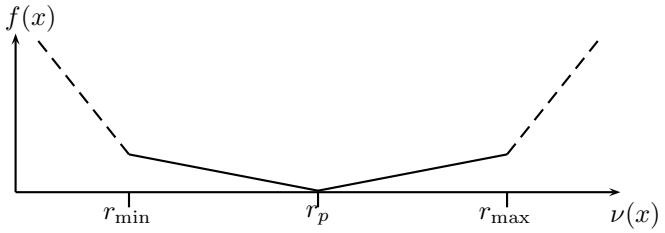


Figure B.1: Schematic of a generic rule composition and objective function terms, with preference term (solid line) and penalties for exceeding min or max (dashed line).

limits of a value, which is calculated from the solution. A value-based rule in the framework has two associated limits, a minimum allowed value $r_{\min}$ and a maximum $r_{\max}$. In addition, such a rule can have a preference r_p that stipulates a preferred value. Deviation from the preference is penalized linearly in the objective function as an objective term.

Rules may be *relaxed* to allow violations to $r_{\min}$ or $r_{\max}$. The violations are penalized in the same way as the preference objectives, but as independent terms with higher α_o weights. This is illustrated in Figure B.1.

Some of the value-based rules are *volume* rules that specify the size or composition of (portions of the) solution as a whole. In the list of rule examples, rules 8–11 fit this description. The volume rules can be used to model the size and composition of the workforce. Volume rules can also provide bounds for constraints that arise in the later stages of workforce scheduling by setting the rule limits appropriately. See for instance Burns and Carter [3] for an application of lower bounds for the "X of Y weekends off" rule. As a result, a large number of volume rules can be in use, to realistically determine the shape of the solution in details.

B.3.3 Rule Modeling

Formally speaking, a rule $r \in \mathcal{R}$ evaluates a value function $\nu(x) \in \mathbb{R}$ on the solution x and the rule is violated if $\nu(x)$ exceeds the specified $r_{\min}$ or $r_{\max}$. The preference term may be calculated from the value function by

$$o(x) = |\nu(x) - r_p|,$$

where r_p is the preferred value of the rule. The penalty function is calculated in a similar way:

$$o(x) = \max\{(\nu(x) - r_{\min}, 0\} + \max\{(r_{\max} - \nu(x), 0\}.$$

A consequence of the rule value function $\nu(x)$ is that it implicitly defines a domain as well. If a rule is specified for a certain contract, only shifts for that contract will contribute to $\nu(x)$. Similarly a rule for a specific weekday will only use shifts on that day in its definition of $\nu(x)$. The domain of each rule is implemented directly in $\nu(x)$, but may be controlled by parameters to the rule. Standard parameters include contract and time interval. When considering time periods, it is useful to consider time intervals that are recurring over a time period, for instance a time period recurring daily or weekly. Such recurrences can be modeled using three parameters, $I = (i_s, i_e, i_p)$. A time period t is in I if $t \geq (i_s \bmod i_p)$ and $t \leq (i_e \bmod i_p)$.

Rules are specified for each qualification, allowing different rules for different parts of the workforce. Otherwise, domains are not explicitly defined for r, as this might restrict the definition or efficiency of the rules to only handle certain types. Instead, each rule may use standard parameter implementations or supply its own implementation.

B.4 Algorithm

In this section, we present the algorithm used to solve the shift design problem. The algorithm is a *simulated annealing*-based local search algorithm, which uses several neighborhoods.

When designing a local search heuristic, a number of concerns must be considered. The time complexity of the neighborhoods should be kept low, and the algorithm should show satisfactory convergence.

These are often contradicting requirements that a good local search algorithm will attempt to balance: Large complex neighborhoods will increase the running time of each individual iteration, but there is time for few iterations during the algorithm run. On the other hand, small simple neighborhoods will be less time consuming, but the expected improvement in quality of a single iteration is likely to be low.

For a real-life setting, where the algorithm is intended for a generalized set of rules and objectives, other concerns must be considered as well. The system should be able to support a large number of rules and objectives, and it should be simple to add support for new rules and objectives as well. New items should be simple to add both during the initial development and as an upgrade to a system already in production.

Finally, it is desirable to have an algorithm that is easy to implement and maintain. This not only reduces the cost of development, but also minimizes the cost of knowledge transfer between operations researchers and software developers.

B.4.1 Algorithm Overview

The local search algorithm uses multiple neighborhoods $N_i : x \to x'$, that each define a transformation of the current solution x solution to a new solution.

The algorithm starts with an initial solution x^0 and iteratively applies one of the neighborhoods to the current solution.

After each iteration, the modified solution x' is evaluated using a *rule engine* that evaluates all hard (i.e. non-relaxed) rules. If all hard rules are satisfied, we use the simulated annealing acceptance criteria, in which the objective function $f(x')$ is evaluated and accepted with probability

$$p = e^{\frac{f(x')-f(x)}{T}}$$

where T is the temperature. If the modified solution is not feasible, or the solution is not accepted, the modified solution is discarded. If accepted, x' replaces x as the current solution for the next iteration. The temperature is updated as $T = T \cdot C$ after $0.05 \cdot D$ iterations. The temperature is updated 100 times, yielding $5 \cdot D$ iterations in total. The initial temperature is set to $T_R \cdot D$. The relative temperature T_R and the cooling factor C are considered parameters, and suitable values will be investigated in Section B.5.3. The remaining values are either fixed or calculated from the size of each individual problem instance $\|D\|_*$.

We use five neighborhoods that each modifies the solution using different "atomic" transformations:

N_1 **Move (translate) Shift:** Randomly select a shift and move the shift to a new random start time.

N_2 **Change Shift Duration:** Randomly select a shift and set a new random duration for the shift. Any breaks made obsolete are removed. Any new breaks are placed randomly.

N_3 **Move Break:** Randomly select a break and give it a new random placement.

N_4 **Create Shift:** Randomly select a qualification, start time, duration, and locations of all breaks.

N_5 **Remove Shift:** Remove a randomly selected shift from the solution.

The neighborhoods chosen are all somewhat intuitive, as they each have a single, unique purpose on their own dimension of the solution. Indeed, similar neighborhoods ware used in the implementations by Dowling [8] (N_1, N_2, N_3), Musliu [17] (variants of N_1, N_2, N_4), and DiGasparo [6] (variants of N_2). The two latter references also consider other neighborhood structures.

When using multiple neighborhoods, there are different ways to select which neighborhood to use in each iteration. The simplest are to select one at random each iteration or use a round robin scheme. We have chosen to use a variant of the adaptive selection strategy of Pisinger and Røpke [18], because our neighborhoods consider different aspects of the solution, and it seems likely that the usefulness of each neighborhood will change during the iterations. Each neighborhood N_i is assigned a weight w_i between 0 and 1, such thay $\sum_{i=1}^{n} w_i = 1$. The selected neigborhood is selected using a roulette wheel principle with probability w_i. In addition, each neighborhood collects a score π_i that sums the achievements of neighborhood N_i. The score is increased by a constant value σ_A if the solution found by the neighborhood is accepted. If the solution has the lowest objective value seen so far, the score is increased by the constant σ_B.

The iterations are grouped into equally sized segments, after which the neighborhood weights are updated to reflect a combination of the previous weight and the earned score. The new weight w_i' is set to

$$w_i' = (1 - \rho)w_i + \rho \cdot \frac{\pi_i}{sel(N_i)}$$

where $sel(N_i)$ is the number of times N_i was selected in the segment. An addtional scaling step ensures that $\sum w_i' = 1$. At the beginning of each new iteration, we set $\pi_i = 1$.

B.4.2 Rules and Neighborhoods

The set of rules enforced by a neighborhood depends on the implementation of the individual neighborhood. Thus, a neighborhood is not required to implement all rules, which makes the implementation of neighborhoods simpler. Rules that are either difficult to implement in a neighborhood or have little chance of being violated by the neighborhood can be disregarded. This decreases the overall complexity of the neighborhood, and increases both running

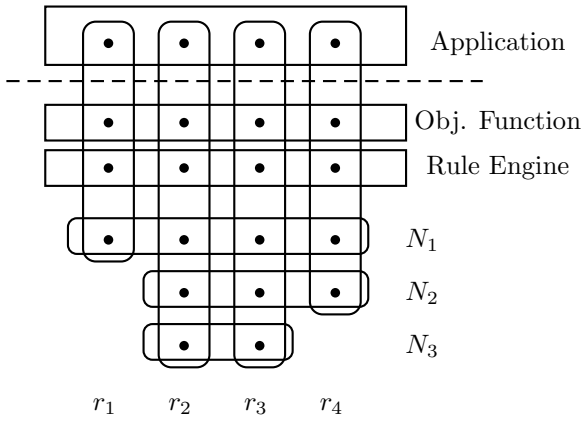


Figure B.2: Schematic representation of the modular structure of rules and neighborhoods.

time and implementation time. An example of the modular structure of rules and neighborhoods is shown in Figure B.2.

Any rule added to a neighborhood will automatically increase the complexity of the neighborhood. This will make the implemented neighborhood more difficult to maintain in a running system, and it will increase the cost of adding additional constraints even further.

As the neighborhood implementations are not required to satisfy all constraints, there is no guarantee that a solution returned by a neighborhood will be feasible. To minimize the amount of time spent in a neighborhood that risk being discarded by the rule engine, the neighborhoods we have initially implemented are small and selects random values rather than searching the entire neighborhood.

Another consequence of the framework is that it is easy to add additional neighborhoods to cooperate with the existing neighborhoods or to replace them. This may be neighborhoods that are able to satisfy a different set of rules or employs more intelligent strategies for selecting new solutions. It may also be composite neighborhoods that combine the actions of two or more "atomic" neighborhoods.

The loosely coupled rule engine means that rules are not explicitly known in the algorithm. For highly constrained problems it may therefore be difficult to ensure that all rules are satisfied throughout the algorithm. For this reason, most rules are relaxed during the early iterations of the algorithm. When a

feasible solution is accepted where no relaxed rules are violated, all relaxed rules are changed to hard rules. From this point, it is no longer possible to violate a rule. This ensures that once a feasible region of the solution space has been found, the algorithm will stay within that region.

B.4.3 Fast Rule Evaluations

To enable both rules and objectives to be evaluated quickly, the concept of delta evaluations is used extensively. In delta evaluations, only the modified parts of the solution are re-calculated, allowing the objectives and rules to calculate only the change (or *delta*) in the value, rather than calculating the value from scratch for the entire solution.

Objectives are in many ways like rules and are treated in the same way. As with rules, they are based on a real-valued function $\nu(x)$, and may be described generally as

$$o(x) = |\nu(x) - o_p|$$

where o_p is the preferred value (or *target*) of the objective. For the objectives that stem from the preference term r_p of a volume-based rule, we can set $o_p = r_p$. For more "pure" objective terms, the target value o_p is often set to 0.

A special case is the coverage objectives, which are calculated by the heuristic of Section B.2.1 rather than a simple value. This also makes the coverage terms the most time consuming. Delta evaluations of the coverage objectives are made from the following simple observations: Only the updated time slots need to be recalculated, since the time slots are calculated independently. Changes to the time slots can be calculated faster, by first checking if overstaffing exists for the changed qualification or if understaffing exists for the most important requirement that can be covered by the qualification. If either exists, these can be updated by adding or subtracting the change. If not, the time slot is calculated again.

B.4.4 Construction Heuristics

To obtain an initial solution we use four approaches. The simplest is to start with an empty solution containing no shifts. We denote this approach C-E.

The second approach is a simple front-loading approach denoted C-FL. Starting from $t = 0$, shifts are added until all workload at t is covered, or no more shifts

starting at t can be added. The front-loading heuristic then considers $t = 1$ and so on, until $t = T$. To keep the complexity of the front-loading heuristic low, all shifts are created with their preferred duration, and breaks are placed in the center of their time windows.

Finally, a dynamic programming heuristic is used to create an initial solution by repeatedly creating sequences of shifts that cover the widest part of the workload. This approach may be considered opposite to C-FL in the sense that it considers workload bottom up, rather than from left to right. The heuristic is used with two different termination criteria, which we denote the *restrictive* approach C-DP-R and the *aggressive* approach C-DP-A. The dynamic programming heuristic is described in details in [5]. In the following we present a short description for the sake of providing a self-contained presentation.

The dynamic programming heuristic computes a non-dominated sequence of shifts for each amount of sequence understaffing (uncovered time units with work) by using shifts from a candidate set S'. To maintain fast solution times, a strict domination scheme scores shifts in the sequence by using a weighted sum $\phi(s)$ of covered work time units, covered idle time units (sequence overstaffing), cost and a measure of skill utilization. The dynamic programming table $\nu[u, t]$ is filled using the recursion in equation (B.1). Recall that l_s and e_s are the length and end time of shift s, respectively.

$$\nu[u,t] = \min_{\substack{s\in S',\\ e_s=t,\\ \forall r:\nu(s)\leq r_{\max}}} \begin{cases} 0 & t = 0, \\ \phi(s) + \nu[u - u(s), t - l_s] & t \geq l_s, u \geq u(s) \\ \infty & \text{otherwise} \end{cases} \tag{B.1}$$

If none of the generated sequences are valid or the ratio of remaining workload to sequence understaffing (overstaffing) is above a threshold parameter $u_{\max}$ ($o_{\max}$), the heuristic terminates. Otherwise, the shifts of the most promising sequence are created, and the heuristic runs again on the incumbent shift set.

The restrictive approach C-DP-R uses the low values $(u_{\max}, o_{\max}) = (0.7, 0.4)$ for the termination parameters and the aggressive uses $(u_{\max}, o_{\max}) = (0.8, 0.6)$. The higher thresholds of the aggressive approach allow the dynamic programming algorithm to generate more sequences but with a higher running time. While the aggresive approach will likely produce more shifts, it is not clear wheter it can do so more efficiently than using more time on the subsequent local search iterations. Both dynamic programming approaches only allow shifts to start every hour, to limit the search space considered in the dynamic programming recursion.

B.5 Computational Results

In this section, we present and discuss computational results for the shift design algorithm. We evaluate the algorithm on real-life data obtained from planning problems in airport ground handling. The instances cover problems from different companies and types of operations, overall spanning a wide variety of instances.

The algorithm should be able to produce good results within a reasonable amount of time in most technical environments. To verify this, all experiments presented in this section were performed on a standard desktop computer with a 3.16Ghz CPU, 4GB of RAM and running Microsoft Windows Vista SP2. The algorithm was implemented in C# .NET 3.5.

We present the data instances in detail in Section B.5.1. The performance of each constructive heuristic is presented in Section B.5.2.

For the parameter tuning, we select two representative instances in order to limit the number of experiments. Section B.5.3 present results of running the algorithm with different parameters and constructive heuristics on the representative instances. From the experiments, we determine the best combination of parameters and constructive heuristic. Finally, we present the results of the algorithm on all instances, using the selected parameters.

B.5.1 Data Instances

Data instances are taken from planning scenarios for real-life operations in airport ground handling. The instances are presented in Table B.1 and covers a variety of different sizes, organizations and types of operation within the field. The scenarios include applications of cargo, operations center manning, passenger services, ramp work and transportation.

The instances are evaluated as received, without further modification. This means that the objective function and rules are considered fixed as the best way to achieve the underlying goal of the planning problem. We assume that where possible, any steps taken to adjust the objective weights or the rule sets have already been taken in order to optimize the performance and convergence of the algorithm.

The instances cover the scenarios also considered in [5]. A more in-depth description is provided there.

Scenario	$\mathcal{R}$	$\mathcal{O}$			D	T
cargo.m.a	205	31	8	5	15294	2016
ops.e.a	30	27	3	5	17887	8064
pax.e.a	59	51	6	10	59329	8064
pax.g.a	12	6	1	2	23962	8064
pax.g.b	12	6	1	2	28286	8064
pax.m.a	389	84	10	9	12796	2016
pax.r.a	72	58	3	8	22326	2016
pax.r.b	108	98	3	12	7442	672
ramp.e.a	222	300	9	15	13354	2688
ramp.e.b	66	55	7	10	59383	8064
ramp.g.a	12	6	1	2	14550	2016
ramp.g.b	12	6	1	2	17420	2016
ramp.m.a	369	56	13	9	3448	2016
ramp.r.a	36	34	1	4	3886	672
trans.o.a	96	74	6	12	25850	2016
trans.o.b	96	74	6	12	39206	2016
	112	60	4	7	22776	3780

Table B.1: Overview of problem instances

B.5.2 Construction Heuristics

In this section, we evaluate the performance of the four constructive heuristics presented in Section B.5.2. An overview of the solutions generated by the four approaches is shown in Table B.2. From the table we see that C-FL is the fastest of the active heuristics with a maximum running time of one second for the largest instance. It also produces the best solution for three of the instance. C-DP-R performs better than C-FL but at a significant increase in running time. It produces the overall best solution on three of the instances with an average running time of 10 seconds. The aggressive heuristic C-DP-A is best for the remaining 10 scenarios, in some cases improving the objective by a factor of 10 compared to C-FL. It is also the slowest heuristic with running times of 15 seconds on average.

B.5.3 Parameters

To determine the best set of parameters for the algorithm, we use two sample instances, `pax.r.a` and `ramp.g.a`. The instances are chosen as representatives, as they have different characteristics without deviating too much from the remaining instances.

Scenario	C-E $f(x)$	C-FL $f(x)$	C-FL Time	C-DP-R $f(x)$	C-DP-R Time	C-DP-A $f(x)$	C-DP-A Time
cargo.m.a	$2.5331 \cdot 10^{05}$	$8.6383 \cdot 10^{04}$	0.1	$\mathbf{7.4102 \cdot 10^{04}}$	0.79	$7.7455 \cdot 10^{04}$	0.73
ops.e.a	$7.7148 \cdot 10^{05}$	$5.8674 \cdot 10^{05}$	0.13	$5.0458 \cdot 10^{05}$	11.14	$\mathbf{4.4930 \cdot 10^{05}}$	18.59
pax.e.a	$2.6046 \cdot 10^{06}$	$1.4369 \cdot 10^{06}$	1.01	$\mathbf{8.2089 \cdot 10^{05}}$	60.13	$1.1079 \cdot 10^{06}$	60.35
pax.g.a	$2.5719 \cdot 10^{10}$	$6.5634 \cdot 10^{09}$	0.08	$1.8123 \cdot 10^{09}$	15.17	$\mathbf{6.2739 \cdot 10^{08}}$	26.15
pax.g.b	$3.5196 \cdot 10^{10}$	$9.0306 \cdot 10^{09}$	0.09	$2.3304 \cdot 10^{09}$	18.46	$\mathbf{8.8673 \cdot 10^{08}}$	29.78
pax.m.a	$4.1104 \cdot 10^{05}$	$2.8273 \cdot 10^{05}$	0.11	$2.8850 \cdot 10^{05}$	1.08	$\mathbf{2.3571 \cdot 10^{05}}$	1.48
pax.r.a	$3.9872 \cdot 10^{06}$	$1.6040 \cdot 10^{06}$	0.22	$3.0652 \cdot 10^{05}$	3.83	$\mathbf{2.4564 \cdot 10^{05}}$	7.27
pax.r.b	$6.8482 \cdot 10^{04}$	$\mathbf{3.0692 \cdot 10^{03}}$	0.22	$5.2705 \cdot 10^{03}$	2.11	$4.1243 \cdot 10^{03}$	3.51
ramp.e.a	$7.2922 \cdot 10^{14}$	$7.2580 \cdot 10^{14}$	0.43	$\mathbf{7.2368 \cdot 10^{14}}$	5.69	$7.2427 \cdot 10^{14}$	6.76
ramp.e.b	$2.1238 \cdot 10^{06}$	$8.4437 \cdot 10^{05}$	0.55	$6.2824 \cdot 10^{05}$	32.98	$\mathbf{2.9066 \cdot 10^{05}}$	59.2
ramp.g.a	$4.7057 \cdot 10^{09}$	$4.4586 \cdot 10^{08}$	0.05	$1.9721 \cdot 10^{08}$	1.61	$\mathbf{9.6253 \cdot 10^{07}}$	2.66
ramp.g.b	$6.6335 \cdot 10^{09}$	$6.2474 \cdot 10^{08}$	0.07	$2.5870 \cdot 10^{08}$	1.86	$\mathbf{1.0783 \cdot 10^{08}}$	3.14
ramp.m.a	$1.4172 \cdot 10^{05}$	$9.7905 \cdot 10^{04}$	0.07	$1.1888 \cdot 10^{05}$	1.01	$\mathbf{9.6541 \cdot 10^{04}}$	1.09
ramp.r.a	$4.1445 \cdot 10^{04}$	$\mathbf{1.5664 \cdot 10^{03}}$	0.05	$3.4031 \cdot 10^{03}$	0.78	$1.9928 \cdot 10^{03}$	1.41
trans.o.a	$1.8656 \cdot 10^{10}$	$1.6646 \cdot 10^{09}$	0.22	$1.5314 \cdot 10^{09}$	3.78	$\mathbf{8.3388 \cdot 10^{08}}$	5.93
trans.o.b	$2.6561 \cdot 10^{10}$	$\mathbf{6.7953 \cdot 10^{08}}$	0.36	$1.8797 \cdot 10^{09}$	8.39	$7.6738 \cdot 10^{08}$	14.93
Average	$4.5584 \cdot 10^{13}$	$4.5364 \cdot 10^{13}$	0.24	$4.5231 \cdot 10^{13}$	10.55	$4.5267 \cdot 10^{13}$	15.19

Table B.2: Objective values and running times for the construction heuristics on all scenarios

We evaluate the performance of the algorithm using each of the four constructive heuristics.

To reduce the number of tests, we limit the search to only two of the parameters, the cooling C and the relative start temperature T_R. The start temperature is set by $T_s = T_r \cdot \sqrt{D}$. This is to scale the temperature to the dimensions of each problem instance and to reflect the quadratic nature of the coverage objective terms, which we assume to be dominant in all instances.

The remaining parameters are fixed at values that have been observed to provide good results during initial testing. The algorithm runs $0.05 \cdot D$ iterations at each temperature step and 100 temperature steps in total. For the adaptive neighborhood selection, the neighborhood scores are updated by $\sigma_A = 1$ for a accepted solution and $\sigma_B = 30$ for obtaining the best solution so far. The new neighborhood weights w_i' are calculated using reaction factor $\rho = 0.3$ and segment size 100.

We test the performance of the algorithm on the scenarios using parameters $C = \{0.93, 0.95, 0.97, 0.99\}$ and $T_R = \{0.25, 0.5, 0.75, 0.1\}$ using all four constructive heuristics.

The results for `ramp.r.a` using the different constructive heuristics are presented in the appendix in Tables B.5, B.6, B.7, and B.8. The tables show the resulting start temperature T_s and end temperature T_e, objective value and running time, as well as resulting overstaffing o, understaffing u, the number of shifts in the resulting shift set $\mathcal{S}$ and the number of rules $\mathcal{R}^+$ that remains violated in the final solution. The results are averages of 3 runs and are presented sorted by objective value.

Generally, the cooling factor 0.99 performs poorly on `pax.r.a` for all constructive heuristics. It seems that the resulting end temperature is too high to settle into a suitable local minimum. In no cases was the algorithm able to satisfy all rules. Best result is using C-E, where 6 relaxed rules remain. However, this solution is worse than any other by orders of magnitude, showing that the best solutions for `pax.r.a` are not found by satisfying all rules. Rather, it seems that solutions with low values for o and u perform best from a practical perspective. Most solution values are between $1 \cdot 10^4$ and $2 \cdot 10^4$, which improves the solutions found by the constructive heuristics alone, as reported in Table B.2. Most solution times are just above two minutes.

Results for applying the local search algorithm to each of the four constructive heuristics on `ramp.g.a` are shown in the appendix Tables B.9, B.10, B.11, and B.12. For this scenario, objective values are higher, most values lie between $2.6 \cdot 10^6$ and $3.6 \cdot 10^6$. All solutions found have no relaxed rules and no understaffing.

The scenario is smaller, with solution times slightly above 30 seconds.

Table B.3 summarizes Tables B.5– B.12 by presenting the objective values of all constructive heuristics for both scenarios. The bottom row shows the average objective value found for each scenario. From this it is seen that the frontloading heuristic C-FL clearly provides the basis for the best solution. This is somewhat surprising, as both C-DP-R and C-DP-A provides better initial solutions. The dynamic programming heuristics provide *dense* packings of shifts to the demand, which may constitute local minima to the subsequent local search algorithm. Experiments show that the extra calculation time used by the dynamic programming heuristics for obtaining better initial solutions is not worthwhile.

The rightmost column of Table B.3 shows a weighted objective ratio of all objective values for each parameter pair. The weighted ratio is calculated as $w = \sum_{i=1}^{n}(x_i/\bar{x}_i)/n$, where $\bar{x}_i$ is the average value of the corresponding column, as seen in the bottom row. The weighted ratio averages the performance for each scenario / heuristic as compared to that combination's average. A ratio of $w = 1$ indicates that a parameter combination is consistently average, while lower values are better. The ratio is preferred to a regular average to minimize the dominance of results for `ramp.g.a`, which have higher objective values. From this ratio it is seen that $(T_r, C) = (1, 0.97)$ performs best with an average ratio of 0.86.

B.5.4 Final Experiments

In this section, we report results of solving all the real-life instances. The algorithm uses the parameters determined in the end of Section B.5.3. Results are generated using the constructive heuristic C-FL, that was found to perform best.

Average results over three runs for all instances are presented in Table B.4. The objective value and percentagewise improvement over the initial objective value is reported, along with running times and the number of iterations. The levels of overstaffing and understaffing are also reported, as well as the number of shifts in the solution and the number of relased rules remaining.

The improvement in objective value is observed to increase for all scenarios. For 7 of the 16 scenarios, the improvement is more than 99%, which indicate that the final solution is better by several orders of magnitude. Only three scenarios have reported improvements of less than 10%, with the single scenario `pax.r.b` obtaining only 0.02% no improvement over the initial solution. For all other scenarios, the improvement of the local search algorithm is significant.

		pax.r.a				ramp.g.a				
T_R	C	C-E	C-FL	C-DP-R	C-DP-A	C-E	C-FL	C-DP-R	C-DP-A	w
0.25	0.93	$1.86 \cdot 10^4$	$1.48 \cdot 10^4$	$1.72 \cdot 10^4$	$1.90 \cdot 10^4$	$3.03 \cdot 10^6$	$2.72 \cdot 10^6$	$3.20 \cdot 10^6$	$3.41 \cdot 10^6$	0.89
0.25	0.95	$1.85 \cdot 10^4$	$1.51 \cdot 10^4$	$1.74 \cdot 10^4$	$1.91 \cdot 10^4$	$3.03 \cdot 10^6$	$2.66 \cdot 10^6$	$3.31 \cdot 10^6$	$3.45 \cdot 10^6$	0.90
0.25	0.97	$1.77 \cdot 10^4$	$1.49 \cdot 10^4$	$1.67 \cdot 10^4$	$1.95 \cdot 10^4$	$3.00 \cdot 10^6$	$2.68 \cdot 10^6$	$3.19 \cdot 10^6$	$3.66 \cdot 10^6$	0.90
0.25	0.99	$1.90 \cdot 10^4$	$1.60 \cdot 10^4$	$1.80 \cdot 10^4$	$1.99 \cdot 10^4$	$3.05 \cdot 10^6$	$2.71 \cdot 10^6$	$3.13 \cdot 10^6$	$3.50 \cdot 10^6$	0.91
0.50	0.93	$1.88 \cdot 10^4$	$1.43 \cdot 10^4$	$1.68 \cdot 10^4$	$1.79 \cdot 10^4$	$3.05 \cdot 10^6$	$2.66 \cdot 10^6$	$3.24 \cdot 10^6$	$3.47 \cdot 10^6$	0.88
0.50	0.95	$1.68 \cdot 10^4$	$1.51 \cdot 10^4$	$1.61 \cdot 10^4$	$1.89 \cdot 10^4$	$3.09 \cdot 10^6$	$2.73 \cdot 10^6$	$3.30 \cdot 10^6$	$3.57 \cdot 10^6$	0.90
0.50	0.97	$1.77 \cdot 10^4$	$1.45 \cdot 10^4$	$1.68 \cdot 10^4$	$1.82 \cdot 10^4$	$2.99 \cdot 10^6$	$2.77 \cdot 10^6$	$3.31 \cdot 10^6$	$3.56 \cdot 10^6$	0.89
0.50	0.99	$2.01 \cdot 10^4$	$1.83 \cdot 10^4$	$1.92 \cdot 10^4$	$2.06 \cdot 10^4$	$3.01 \cdot 10^6$	$2.70 \cdot 10^6$	$3.37 \cdot 10^6$	$3.60 \cdot 10^6$	0.96
0.75	0.93	$1.72 \cdot 10^4$	$1.43 \cdot 10^4$	$1.67 \cdot 10^4$	$1.79 \cdot 10^4$	$2.97 \cdot 10^6$	$2.80 \cdot 10^6$	$3.27 \cdot 10^6$	$3.48 \cdot 10^6$	0.88
0.75	0.95	$1.67 \cdot 10^4$	$1.43 \cdot 10^4$	$1.59 \cdot 10^4$	$1.77 \cdot 10^4$	$3.10 \cdot 10^6$	$2.70 \cdot 10^6$	$3.21 \cdot 10^6$	$3.57 \cdot 10^6$	0.88
0.75	0.97	$1.23 \cdot 10^6$	$1.46 \cdot 10^4$	$1.62 \cdot 10^4$	$1.60 \cdot 10^4$	$3.11 \cdot 10^6$	$2.71 \cdot 10^6$	$3.27 \cdot 10^6$	$3.45 \cdot 10^6$	2.48
0.75	0.99	$2.21 \cdot 10^4$	$2.11 \cdot 10^4$	$2.22 \cdot 10^4$	$2.28 \cdot 10^4$	$3.02 \cdot 10^6$	$2.74 \cdot 10^6$	$3.22 \cdot 10^6$	$3.56 \cdot 10^6$	1.01
1.00	0.93	$1.66 \cdot 10^4$	$1.43 \cdot 10^4$	$1.60 \cdot 10^4$	$1.68 \cdot 10^4$	$3.06 \cdot 10^6$	$2.74 \cdot 10^6$	$3.24 \cdot 10^6$	$3.48 \cdot 10^6$	0.87
1.00	0.95	$1.66 \cdot 10^4$	$1.50 \cdot 10^4$	$1.54 \cdot 10^4$	$1.70 \cdot 10^4$	$3.14 \cdot 10^6$	$2.80 \cdot 10^6$	$3.31 \cdot 10^6$	$3.58 \cdot 10^6$	0.88
1.00	0.97	$1.65 \cdot 10^4$	$1.44 \cdot 10^4$	$1.58 \cdot 10^4$	$1.59 \cdot 10^4$	$3.05 \cdot 10^6$	$2.77 \cdot 10^6$	$3.28 \cdot 10^6$	$3.43 \cdot 10^6$	0.86
1.00	0.99	$2.53 \cdot 10^4$	$2.39 \cdot 10^4$	$2.46 \cdot 10^4$	$2.53 \cdot 10^4$	$3.07 \cdot 10^6$	$2.69 \cdot 10^6$	$3.18 \cdot 10^6$	$3.48 \cdot 10^6$	1.07
0.63	0.96	$9.40 \cdot 10^4$	$1.59 \cdot 10^4$	$1.76 \cdot 10^4$	$1.89 \cdot 10^4$	$3.05 \cdot 10^6$	$2.72 \cdot 10^6$	$3.25 \cdot 10^6$	$3.52 \cdot 10^6$	1.01

Table B.3: Overview of objective values for combinations of parameters on sample instances `pax.r.a` and `ramp.g.a` using four construction heuristics. Rightmost column and bottom row show average values.

Scenario	$f(x^*)$	Imp.	Time (s)	Iters	o	u	$\mathcal{S}$	$\mathcal{R}^+$
`cargo.m.a`	$7.5521 \cdot 10^{04}$	12.57%	114.8	76400	1295	6269	107	1
`ops.e.a`	$2.9254 \cdot 10^{05}$	50.14%	61.17	89400	50244	2850	749	4
`pax.e.a`	$9.9756 \cdot 10^{05}$	30.58%	401.74	296600	107397	8090	1888	14
`pax.g.a`	$9.6546 \cdot 10^{06}$	99.85%	53.83	119800	13615	0	505	0
`pax.g.b`	$1.1818 \cdot 10^{07}$	99.87%	63.98	141400	15607	0	592	0
`pax.m.a`	$1.8771 \cdot 10^{05}$	33.61%	108.85	63900	10955	5807	186	18
`pax.r.a`	$1.4152 \cdot 10^{04}$	99.12%	121.1	111600	506	72	387	8
`pax.r.b`	$3.0686 \cdot 10^{03}$	0.02%	30.01	37200	3444	133	478	0
`ramp.e.a`	$6.6766 \cdot 10^{14}$	8.01%	94.5	66700	5218	4309	616	12
`ramp.e.b`	$6.0009 \cdot 10^{05}$	28.93%	461.25	296900	60407	10881	1386	11
`ramp.g.a`	$2.7486 \cdot 10^{06}$	99.38%	29.34	72700	8356	0	326	0
`ramp.g.b`	$3.4089 \cdot 10^{06}$	99.45%	35.53	87100	9201	0	376	0
`ramp.m.a`	$2.5813 \cdot 10^{04}$	73.63%	39.21	17200	9808	682	132	1
`ramp.r.a`	$1.5130 \cdot 10^{03}$	3.41%	9.02	19400	2267	66	273	0
`trans.o.a`	$1.2433 \cdot 10^{07}$	99.25%	130.95	129200	27324	0	745	12
`trans.o.b`	$1.3231 \cdot 10^{07}$	98.05%	193.5	196000	35766	0	1041	12
	$4.1729 \cdot 10^{13}$	58.49	121.8	113843	22588	2447	611	5

Table B.4: Solution statistics for all instances.

Although the local search algorithm can increase the initial solution significantly, it still benefits from a construction heuristic. This is evidenced by Table B.3, where the empty heuristic C-E performs worse than the chosen heuristic C-FL on 31 of the 32 presented cases.

The majority of the scenarios are solved in two minutes or less. The only scenarios that significantly exceed two minutes are `trans.o.b` (3.2 minutes), `pax.e.a` (6 minutes), and `ramp.e.b` (7 minutes). The number of iterations used varies according to problem size, but does not directly influence running time. The average number of iterations per second is between 440 and 2500, with the 8 fastest scenarios being over 1000 iterations per second.

The number of violated rules $\mathcal{R}^+$ remaining in the solutions varies. Six scenarios are consistently solved with no violated rules. Two additional scenarios are solved with only a single violated rule. On average, 5 violated rules remain, which is less than 5% of the 112 rules in the scenarios on average. It should also be noted that for some scenarios, the objective function may prefer better coverage over satisfying relaxed rules. This appears to be the case for `ramp.r.a`, as seen in Table B.5.

B.6 Conclusions and Future Work

In this paper we have presented a local search algorithm for the shift design problem. The algorithm uses a loosely coupled rule engine to provide a large degree of flexibility in modeling and solving shift design problems arising in different organizations or operations.

A modular design of the metaheuristic allows for easy expansions to cover new rules or objectives, either by updating the existing neighborhoods, or by adding new competitive neighborhoods covering new rules.

Experiments have been performed on real-life instances covering a wide variety of scenarios occurring in airport ground handling, using four different construction heuristics followed by the local search algorithm. The most time-consuming heuristics provided the best results, but in the majority of cases the subsequent local search algorithm was able to improve the result significantly. Interestingly, the best suited constructive heuristic did not perform best on its own, but rather created solutions which were not located in a deep local minimum. This provided the algorithm better conditions for improving the initial solution further. In addition, the experiments show that the presented algorithm presented is capable of successfully solving such problems and handling a varied set of constraints occurring in this area, within short running times. The algorithm is shown to generate good results on a set of instances with large variations in size, and the number of objectives and rules.

The neighborhoods presented in this paper cover atomic transitions that are natural to shift design problems. It will be interesting to expand the knowledge of the framework through new and more complex neighborhood transitions and selection strategies.

The adaptive neighborhood scoring method used to select neighborhoods evaluates all neighborhoods using the same criteria, and thus implicitly assumes that all neighborhoods can have the same impact on the solution. Using the framework it is difficult to accommodate both neighborhoods which modify the solution slightly but have a high chance of acceptance and neighborhoods which modify the solution more extensively, but have a lower chance of acceptance. The definition of Adaptive Large Neighborhood Search by Pisinger and Røpke [18] (which inspired our implementation) explicitly defines all neighborhoods to have the same size. Thompson [21] integrated a “finetune” step as a final part of the neighborhood operation, thus explicitly separating large and small updates to the solution. Mladenovic and Hansen [16] propose Variable Neighborhood Search which explicitly defines a transition between a set of growing neighborhoods. Both approaches have the disadvantage that the

neighborhood operations must be determined a priori, which conflicts with the flexibility of an arbitrary number of neighborhoods considered here. A method for scoring neighborhoods of arbitrary size would be interesting for further study.

The need for solving generic problems with a varying set of constraints exists in many areas of planning and workforce scheduling. In particular, it will be interesting to apply the framework proposed in this paper to instances of other organizations and areas, and to other planning problems arising in workforce planning and scheduling.

B.7 Additional Tables

Scenario	T_R	C	T_s	T_e	$f(x^*)$	Time (s)	o	u	$\mathcal{S}$	$\mathcal{R}^+$
pax.r.a	1.00	0.97	149.42	7.11	$1.6550 \cdot 10^{04}$	119.62	559	85	403	9
pax.r.a	1.00	0.93	149.42	0.11	$1.6582 \cdot 10^{04}$	123.13	590	64	398	10
pax.r.a	1.00	0.95	149.42	0.88	$1.6595 \cdot 10^{04}$	122.26	565	77	398	10
pax.r.a	0.75	0.95	112.06	0.66	$1.6656 \cdot 10^{04}$	121.97	579	59	403	10
pax.r.a	0.50	0.95	74.71	0.44	$1.6769 \cdot 10^{04}$	123.4	594	64	403	10
pax.r.a	0.75	0.93	112.06	0.08	$1.7227 \cdot 10^{04}$	121.19	628	66	402	10
pax.r.a	0.50	0.97	74.71	3.55	$1.7668 \cdot 10^{04}$	120.59	667	63	402	10
pax.r.a	0.25	0.97	37.35	1.78	$1.7740 \cdot 10^{04}$	121.95	698	70	399	10
pax.r.a	0.25	0.95	37.35	0.22	$1.8523 \cdot 10^{04}$	122.99	702	82	398	10
pax.r.a	0.25	0.93	37.35	0.03	$1.8624 \cdot 10^{04}$	123.39	684	82	400	10
pax.r.a	0.50	0.93	74.71	0.05	$1.8801 \cdot 10^{04}$	123.07	729	73	401	10
pax.r.a	0.25	0.99	37.35	13.67	$1.9036 \cdot 10^{04}$	121.26	724	110	401	10
pax.r.a	0.50	0.99	74.71	27.35	$2.0076 \cdot 10^{04}$	117.72	744	186	402	10
pax.r.a	0.75	0.99	112.06	41.02	$2.2087 \cdot 10^{04}$	118.78	858	228	403	9
pax.r.a	1.00	0.99	149.42	54.69	$2.5258 \cdot 10^{04}$	119.57	942	272	400	9
pax.r.a	0.75	0.97	112.06	5.33	$1.2252 \cdot 10^{06}$	99.41	402	7078	272	6

Table B.5: Average results over 3 runs with different parameters for `pax.r.a` with constructive heuristic C-E.

Scenario	T_R	C	T_s	T_e	$f(x^*)$	Time (s)	o	u	$\mathcal{S}$	$\mathcal{R}^+$
pax.r.a	0.75	0.93	112.06	0.08	$1.4276 \cdot 10^{04}$	123.36	552	48	392	8
pax.r.a	0.50	0.93	74.71	0.05	$1.4329 \cdot 10^{04}$	120.85	557	53	390	8
pax.r.a	0.75	0.95	112.06	0.66	$1.4338 \cdot 10^{04}$	119.28	550	58	390	8
pax.r.a	1.00	0.93	149.42	0.11	$1.4345 \cdot 10^{04}$	120.78	581	45	391	10
pax.r.a	1.00	0.97	149.42	7.11	$1.4376 \cdot 10^{04}$	118.93	550	60	391	9
pax.r.a	0.50	0.97	74.71	3.55	$1.4488 \cdot 10^{04}$	117.89	590	60	385	9
pax.r.a	0.75	0.97	112.06	5.33	$1.4624 \cdot 10^{04}$	117.88	557	63	388	9
pax.r.a	0.25	0.93	37.35	0.03	$1.4756 \cdot 10^{04}$	121.21	632	46	389	9
pax.r.a	0.25	0.97	37.35	1.78	$1.4897 \cdot 10^{04}$	119.19	621	47	392	9
pax.r.a	1.00	0.95	149.42	0.88	$1.4957 \cdot 10^{04}$	120.03	619	47	390	9
pax.r.a	0.50	0.95	74.71	0.44	$1.5063 \cdot 10^{04}$	119.14	652	46	389	9
pax.r.a	0.25	0.95	37.35	0.22	$1.5090 \cdot 10^{04}$	120.94	605	65	386	8
pax.r.a	0.25	0.99	37.35	13.67	$1.5988 \cdot 10^{04}$	118.94	649	97	388	9
pax.r.a	0.50	0.99	74.71	27.35	$1.8347 \cdot 10^{04}$	117.02	718	182	385	9
pax.r.a	0.75	0.99	112.06	41.02	$2.1135 \cdot 10^{04}$	119.08	881	193	392	9
pax.r.a	1.00	0.99	149.42	54.69	$2.3891 \cdot 10^{04}$	120.35	1045	247	390	9

Table B.6: Average results over 3 runs with different parameters for `pax.r.a` with constructive heuristic C-FL.

Scenario	T_R	C	T_s	T_e	$f(x^*)$	Time (s)	o	u	$\mathcal{S}$	$\mathcal{R}^+$
pax.r.a	1.00	0.95	149.42	0.88	$1.5426 \cdot 10^{04}$	126	566	64	393	9
pax.r.a	1.00	0.97	149.42	7.11	$1.5838 \cdot 10^{04}$	124.55	579	83	390	9
pax.r.a	0.75	0.95	112.06	0.66	$1.5935 \cdot 10^{04}$	122.49	597	63	404	11
pax.r.a	1.00	0.93	149.42	0.11	$1.5990 \cdot 10^{04}$	124.01	634	58	397	10
pax.r.a	0.50	0.95	74.71	0.44	$1.6062 \cdot 10^{04}$	122.58	631	57	417	12
pax.r.a	0.75	0.97	112.06	5.33	$1.6154 \cdot 10^{04}$	123.75	622	78	394	10
pax.r.a	0.75	0.93	112.06	0.08	$1.6674 \cdot 10^{04}$	122.9	658	64	408	10
pax.r.a	0.25	0.97	37.35	1.78	$1.6685 \cdot 10^{04}$	121.01	634	80	420	12
pax.r.a	0.50	0.93	74.71	0.05	$1.6806 \cdot 10^{04}$	123.52	617	73	424	11
pax.r.a	0.50	0.97	74.71	3.55	$1.6822 \cdot 10^{04}$	122.32	651	77	402	11
pax.r.a	0.25	0.93	37.35	0.03	$1.7227 \cdot 10^{04}$	121.65	658	74	433	13
pax.r.a	0.25	0.95	37.35	0.22	$1.7365 \cdot 10^{04}$	120.1	668	66	428	12
pax.r.a	0.25	0.99	37.35	13.67	$1.7998 \cdot 10^{04}$	119.25	707	117	421	12
pax.r.a	0.50	0.99	74.71	27.35	$1.9189 \cdot 10^{04}$	121.56	744	182	396	9
pax.r.a	0.75	0.99	112.06	41.02	$2.2211 \cdot 10^{04}$	123.89	919	223	392	9
pax.r.a	1.00	0.99	149.42	54.69	$2.4586 \cdot 10^{04}$	126.74	1030	254	393	8

Table B.7: Average results over 3 runs with different parameters for `pax.r.a` with constructive heuristic C-DP-R.

Scenario	T_R	C	T_s	T_e	$f(x^*)$	Time (s)	o	u	$\mathcal{S}$	$\mathcal{R}^+$
pax.r.a	1.00	0.97	149.42	7.11	$1.5924 \cdot 10^{04}$	131.82	576	74	401	9
pax.r.a	0.75	0.97	112.06	5.33	$1.6007 \cdot 10^{04}$	131.04	597	59	408	9
pax.r.a	1.00	0.93	149.42	0.11	$1.6811 \cdot 10^{04}$	131.81	646	60	413	9
pax.r.a	1.00	0.95	149.42	0.88	$1.7035 \cdot 10^{04}$	133.65	641	67	398	9
pax.r.a	0.75	0.95	112.06	0.66	$1.7692 \cdot 10^{04}$	129.54	641	67	427	10
pax.r.a	0.50	0.93	74.71	0.05	$1.7852 \cdot 10^{04}$	128.03	676	58	451	12
pax.r.a	0.75	0.93	112.06	0.08	$1.7922 \cdot 10^{04}$	131.57	691	61	423	10
pax.r.a	0.50	0.97	74.71	3.55	$1.8225 \cdot 10^{04}$	128.59	697	79	425	10
pax.r.a	0.50	0.95	74.71	0.44	$1.8920 \cdot 10^{04}$	127.9	745	63	448	12
pax.r.a	0.25	0.93	37.35	0.03	$1.9014 \cdot 10^{04}$	125.65	689	75	469	12
pax.r.a	0.25	0.95	37.35	0.22	$1.9149 \cdot 10^{04}$	127.12	659	87	471	14
pax.r.a	0.25	0.97	37.35	1.78	$1.9536 \cdot 10^{04}$	125.15	698	70	466	12
pax.r.a	0.25	0.99	37.35	13.67	$1.9851 \cdot 10^{04}$	125.7	740	116	447	12
pax.r.a	0.50	0.99	74.71	27.35	$2.0647 \cdot 10^{04}$	129.37	803	205	404	9
pax.r.a	0.75	0.99	112.06	41.02	$2.2795 \cdot 10^{04}$	133.1	971	217	392	8
pax.r.a	1.00	0.99	149.42	54.69	$2.5315 \cdot 10^{04}$	133.18	995	271	399	9

Table B.8: Average results over 3 runs with different parameters for `pax.r.a` with constructive heuristic C-DP-A.

Scenario	T_R	C	T_s	T_e	$f(x^*)$	Time (s)	o	u	$\mathcal{S}$	$\mathcal{R}^+$
ramp.g.a	0.75	0.93	90.47	0.06	$2.9663 \cdot 10^{06}$	30.45	8544	0	317	0
ramp.g.a	0.50	0.97	60.31	2.87	$2.9929 \cdot 10^{06}$	30.72	8427	0	320	0
ramp.g.a	0.25	0.97	30.16	1.43	$2.9983 \cdot 10^{06}$	30.47	8530	0	318	0
ramp.g.a	0.50	0.99	60.31	22.08	$3.0120 \cdot 10^{06}$	30.53	8510	0	318	0
ramp.g.a	0.75	0.99	90.47	33.11	$3.0223 \cdot 10^{06}$	30.4	8552	0	319	0
ramp.g.a	0.25	0.95	30.16	0.18	$3.0312 \cdot 10^{06}$	30.62	8537	0	320	0
ramp.g.a	0.25	0.93	30.16	0.02	$3.0334 \cdot 10^{06}$	30.67	8610	0	320	0
ramp.g.a	1.00	0.97	120.62	5.74	$3.0473 \cdot 10^{06}$	30.48	8597	0	321	0
ramp.g.a	0.50	0.93	60.31	0.04	$3.0521 \cdot 10^{06}$	30.49	8506	0	321	0
ramp.g.a	0.25	0.99	30.16	11.04	$3.0548 \cdot 10^{06}$	30.62	8516	0	319	0
ramp.g.a	1.00	0.93	120.62	0.09	$3.0647 \cdot 10^{06}$	30.54	8497	0	323	0
ramp.g.a	1.00	0.99	120.62	44.15	$3.0716 \cdot 10^{06}$	30.22	8627	0	321	0
ramp.g.a	0.50	0.95	60.31	0.36	$3.0854 \cdot 10^{06}$	30.39	8592	0	322	0
ramp.g.a	0.75	0.95	90.47	0.54	$3.0971 \cdot 10^{06}$	30.71	8536	0	320	0
ramp.g.a	0.75	0.97	90.47	4.30	$3.1076 \cdot 10^{06}$	30.63	8741	0	323	0
ramp.g.a	1.00	0.95	120.62	0.71	$3.1425 \cdot 10^{06}$	30.77	8613	0	322	0

Table B.9: Average results over 3 runs with different parameters for `ramp.g.a` with constructive heuristic C-E.

Scenario	T_R	C	T_s	T_e	$f(x^*)$	Time (s)	o	u	$\mathcal{S}$	$\mathcal{R}^+$
ramp.g.a	0.50	0.93	60.31	0.04	$2.6586 \cdot 10^{06}$	28.92	8362	0	325	0
ramp.g.a	0.25	0.95	30.16	0.18	$2.6628 \cdot 10^{06}$	29.01	8231	0	318	0
ramp.g.a	0.25	0.97	30.16	1.43	$2.6825 \cdot 10^{06}$	28.9	8366	0	322	0
ramp.g.a	1.00	0.99	120.62	44.15	$2.6876 \cdot 10^{06}$	28.97	8287	0	320	0
ramp.g.a	0.50	0.99	60.31	22.08	$2.6951 \cdot 10^{06}$	28.68	8394	0	322	0
ramp.g.a	0.75	0.95	90.47	0.54	$2.6985 \cdot 10^{06}$	28.98	8317	0	320	0
ramp.g.a	0.75	0.97	90.47	4.30	$2.7115 \cdot 10^{06}$	28.88	8440	0	325	0
ramp.g.a	0.25	0.99	30.16	11.04	$2.7144 \cdot 10^{06}$	28.8	8254	0	322	0
ramp.g.a	0.25	0.93	30.16	0.02	$2.7211 \cdot 10^{06}$	28.96	8378	0	324	0
ramp.g.a	0.50	0.95	60.31	0.36	$2.7299 \cdot 10^{06}$	28.71	8373	0	326	0
ramp.g.a	0.75	0.99	90.47	33.11	$2.7366 \cdot 10^{06}$	28.61	8415	0	323	0
ramp.g.a	1.00	0.93	120.62	0.09	$2.7383 \cdot 10^{06}$	28.88	8314	0	325	0
ramp.g.a	1.00	0.97	120.62	5.74	$2.7665 \cdot 10^{06}$	28.64	8417	0	327	0
ramp.g.a	0.50	0.97	60.31	2.87	$2.7702 \cdot 10^{06}$	28.87	8375	0	323	0
ramp.g.a	0.75	0.93	90.47	0.06	$2.7952 \cdot 10^{06}$	28.76	8409	0	324	0
ramp.g.a	1.00	0.95	120.62	0.71	$2.8018 \cdot 10^{06}$	28.47	8404	0	325	0

Table B.10: Average results over 3 runs with different parameters for `ramp.g.a` with constructive heuristic C-FL.

Scenario	T_R	C	T_s	T_e	$f(x^*)$	Time (s)	o	u	$\mathcal{S}$	$\mathcal{R}^+$
ramp.g.a	0.25	0.99	30.16	11.04	$3.1298 \cdot 10^{06}$	34.62	7652	0	321	0
ramp.g.a	1.00	0.99	120.62	44.15	$3.1812 \cdot 10^{06}$	34.43	7729	0	323	0
ramp.g.a	0.25	0.97	30.16	1.43	$3.1878 \cdot 10^{06}$	34.56	7694	0	326	0
ramp.g.a	0.25	0.93	30.16	0.02	$3.2020 \cdot 10^{06}$	34.71	7695	0	326	0
ramp.g.a	0.75	0.95	90.47	0.54	$3.2149 \cdot 10^{06}$	34.35	7781	0	325	0
ramp.g.a	0.75	0.99	90.47	33.11	$3.2196 \cdot 10^{06}$	34.68	7702	0	323	0
ramp.g.a	0.50	0.93	60.31	0.04	$3.2368 \cdot 10^{06}$	34.56	7769	0	325	0
ramp.g.a	1.00	0.93	120.62	0.09	$3.2436 \cdot 10^{06}$	34.4	7726	0	325	0
ramp.g.a	0.75	0.93	90.47	0.06	$3.2659 \cdot 10^{06}$	34.58	7738	0	323	0
ramp.g.a	0.75	0.97	90.47	4.30	$3.2729 \cdot 10^{06}$	34.56	7822	0	327	0
ramp.g.a	1.00	0.97	120.62	5.74	$3.2766 \cdot 10^{06}$	34.59	7897	0	325	0
ramp.g.a	0.50	0.95	60.31	0.36	$3.2966 \cdot 10^{06}$	34.53	7878	0	327	0
ramp.g.a	0.50	0.97	60.31	2.87	$3.3124 \cdot 10^{06}$	34.53	7862	0	325	0
ramp.g.a	0.25	0.95	30.16	0.18	$3.3124 \cdot 10^{06}$	34.8	7864	0	326	0
ramp.g.a	1.00	0.95	120.62	0.71	$3.3136 \cdot 10^{06}$	34.52	7818	0	325	0
ramp.g.a	0.50	0.99	60.31	22.08	$3.3724 \cdot 10^{06}$	34.37	7933	0	328	0

Table B.11: Average results over 3 runs with different parameters for `ramp.g.a` with constructive heuristic C-DP-R.

Scenario	T_R	C	T_s	T_e	$f(x^*)$	Time (s)	o	u	$\mathcal{S}$	$\mathcal{R}^+$
ramp.g.a	0.25	0.93	30.16	0.02	$3.4137 \cdot 10^{06}$	36.25	7879	0	334	0
ramp.g.a	1.00	0.97	120.62	5.74	$3.4253 \cdot 10^{06}$	36.22	7924	0	331	0
ramp.g.a	0.25	0.95	30.16	0.18	$3.4493 \cdot 10^{06}$	36.05	7924	0	332	0
ramp.g.a	0.75	0.97	90.47	4.30	$3.4510 \cdot 10^{06}$	35.92	7966	0	334	0
ramp.g.a	0.50	0.93	60.31	0.04	$3.4657 \cdot 10^{06}$	36.17	7912	0	334	0
ramp.g.a	0.75	0.93	90.47	0.06	$3.4750 \cdot 10^{06}$	36.15	7971	0	336	0
ramp.g.a	1.00	0.93	120.62	0.09	$3.4779 \cdot 10^{06}$	36.14	7961	0	334	0
ramp.g.a	1.00	0.99	120.62	44.15	$3.4847 \cdot 10^{06}$	36.06	7922	0	333	0
ramp.g.a	0.25	0.99	30.16	11.04	$3.4951 \cdot 10^{06}$	36.26	7997	0	334	0
ramp.g.a	0.75	0.99	90.47	33.11	$3.5579 \cdot 10^{06}$	35.78	8088	0	338	0
ramp.g.a	0.50	0.97	60.31	2.87	$3.5644 \cdot 10^{06}$	36.13	8132	0	337	0
ramp.g.a	0.50	0.95	60.31	0.36	$3.5695 \cdot 10^{06}$	35.83	8090	0	336	0
ramp.g.a	0.75	0.95	90.47	0.54	$3.5706 \cdot 10^{06}$	35.97	8051	0	337	0
ramp.g.a	1.00	0.95	120.62	0.71	$3.5823 \cdot 10^{06}$	36.02	8076	0	334	0
ramp.g.a	0.50	0.99	60.31	22.08	$3.6030 \cdot 10^{06}$	36.06	8152	0	336	0
ramp.g.a	0.25	0.97	30.16	1.43	$3.6551 \cdot 10^{06}$	36.26	8181	0	336	0

Table B.12: Average results over 3 runs with different parameters for `ramp.g.a` with constructive heuristic C-DP-A.

References

[1] S. E. Bechtold and M. J. Brusco. A microcomputer-based heuristic for tour scheduling of a mixed workforce. *Computers and Operations Research*, 21(9):1001–1009, 1994.

[2] M. J. Brusco, L. W. Jacobs, R. J. Bongiorno, D. V. Lyons, and B. Tang. Improving personnel scheduling at airline stations. *Operations Research*, 43(5):741–751 and 172029, 1995.

[3] R. Burns and M. Carter. Work force size and single shift schedules with variable demands. *Management Science*, pages 599–607, 1985.

[4] S. C. Chu. Generating, scheduling and rostering of shift crew-duties: Applications at the hong kong international airport. *European Journal of Operational Research*, 177(3):1764–1778, 2007.

[5] T. Clausen. A dynamic programming-based heuristic for the shift design problem in airport ground handling. Technical Report 7.2010, Technical University of Denmark, Department of Management Engineering, Operations Research, Produktionstorvet, DK-2800 Kgs. Lyngby, 2010.

[6] L. Di Gaspero, J. Gärtner, G. Kortsarz, N. Musliu, A. Schaerf, and W. Slany. The minimum shift design problem. *Annals of Operations Research*, 155(1):79–105, 2007.

[7] A. Dohn, A. Mason, and D. Ryan. A generic approach to nurse rostering. Technical Report 5.2010, Technical University of Denmark, Department of Management Engineering, Operations Research, Produktionstorvet, DK-2800 Kgs. Lyngby, 2010.

[8] D. Dowling, M. Krishnamoorthy, H. Mackenzie, and D. Sier. Staff rostering at a large international airport. *Annals of Operations Research*, 72(0):125–147, 1997.

[9] A. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3–27, 2004.

[10] J. Herbers. *Models and Algorithms for Ground Staff Scheduling On Airports*. Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, Faculty of Mathematics, Computer Science and Natural Sciences, 2005.

[11] S. Ho and J. Leung. Solving a manpower scheduling problem for airline catering using metaheuristics. *European Journal of Operational Research*, 202(3):903–921, 2010.

[12] R. Hung. Single-shift off-day scheduling of a hierarchical workforce with variable demands. *European Journal of Operational Research*, 78(1):49–57, 1994.

[13] N. Kohl and S. E. Karish. Airline crew rostering: Problem types, modeling, and optimization. *Annals of Operations Research*, 127:223–257, 2004.

[14] H. C. Lau. On the complexity of manpower shift scheduling. *Computers & Operations Research*, 23(1):93–102, 1996.

[15] A. J. Mason, D. M. Ryan, and D. M. Panton. Integrated simulation, heuristic and optimisation approaches to staff scheduling. *Operations Research*, 46(2):161–175, 1998.

[16] N. Mladenovic and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

[17] N. Musliu, A. Schaerf, and W. Slany. Local search for shift design. *European Journal of Operational Research*, 153(1):51–64, 2004.

[18] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, 2007.

[19] C. Quimper and L. Rousseau. A large neighbourhood search approach to the multi-activity shift scheduling problem. *Journal of Heuristics*, 16(3):373–392, 2010.

[20] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. *Principles and Practice of Constraint Programming-CP98*, pages 417–431, 1998.

[21] G. Thompson. A simulated-annealing heuristic for shift scheduling using non-continuously available employees. *Computers & Operations Research*, 23(3):275–288, 1996.

[22] J. M. Tien and A. Kamiyama. On manpower scheduling algorithms. *SIAM Review*, 24(3):275–287, 1982.

[23] P. Van Hentenryck and L. Michel. *Constraint-based local search*. The MIT Press, 2009.

Paper C

Dynamic Routing of Short Transfer Baggage

Tommy Clausen and David Pisinger

Submitted to Transportation Research Part E

Dynamic Routing of Short Transfer Baggage

Tommy Clausen and David Pisinger

We consider a variant of the Vehicle Routing Problem that arises in airports when transporting baggage for passengers with connecting flights. Each bag can be delivered in two locations with disjunctive time windows. The task is to define multiple trips for the vehicles in order to deliver bags that arrive continuously during the day. We present an IP model of the problem and describe the problem as a case study from a real life setting. We present a weighted greedy algorithm for dispatching vehicles that works in an dynamic context, meaning that it only considers bags available at the time of dispatch. Computational results are presented for real-life passenger data with stochastic bag arrival times and travel times. The results indicate that the algorithm is able to dispatch the baggage considerably better than the manual delivery plans reported in the case study, and due to its fast running times, the algorithm is suitable for dynamic dispatching. Investigations on the impact of uncertainty and fleet size make it possible to support a trade-off between fleet size and expected service level.

C.1 Introduction

We consider the problem of transporting baggage for passengers with connecting flights in an airport. Connecting flight passengers are passengers that arrive in the airport on an inbound flight and depart on an outbound flight within a brief period of time. The 'transfer baggage' carried by these passengers is not directed to the arrival halls like regular baggage. Instead, it must be transported separately, in order to intercept the baggage for the departing flight.

The process of picking up and and delivering connecting baggage may vary from airport to airport. The process is ruled by the airport's infrastructure and regulations, as well as the subsidiary contracts negotiated by the baggage handling companies. In this paper we consider a setup as follows: The company handling the connecting baggage has a number of baggage sorting and dispatch

terminals into which baggage are brought by the baggage handlers servicing the inbound flights. The company is then responsible for transporting the bags to either the outbound flight or to the baggage handling stations used for the other bags corresponding to the flight. Delivering to the handling stations is only possible if the bags are delivered before the regular bags are taken from the station to the aircraft. See Figure C.1 for an illustration.

To transport the bags, the handling company operates a fleet of homogeneous vehicles, that are dispatched repeatedly throughout the day. Each trip starts at the baggage dispatch facility, visits a number of flights and/or baggage handling stations, and returns to the dispatch facility. When a vehicle returns from a trip, it is either assigned to a new route immediately, or instructed to wait if no bags are currently available.

The problem is to plan the routes for the vehicles such that each bag is either delivered directly to the flight, or to the baggage station, respecting the time windows of each. The objective is to minimize the number of bags which do not catch the flight.

We present an IP-model of the static problem where all bags and departures are known in advance. We then consider the dynamic problem and present a weighted greedy algorithm which only considers bags available at the time of dispatch. Computational results are presented for real-life passenger data with stochastic bag arrival times and travel times. Based on the generated scenarios, it seems that the presented algorithm is able to achieve the same level of service or better with less than half the current fleet size even when considering a substantial level of uncertainty. Studying a large number of strategies for dispatching the bags, experimental results indicate that for the considered problem, risk willingness pays off, also when there is a large degree of uncertainty.

C.1.1 Related Work

The *Short transfer baggage routing problem* (STBRP) may be treated as an offline or an online problem depending on the settings. In the offline version, it is assumed that all input data are known in advance. In the online (or dynamic) version of the problem the input is fed to the algorithm as it becomes available, so the algorithm does not have the entire input available from the start. Hence, the online algorithm is forced to make decisions that may later turn out not to be optimal in an overall sense.

The *offline* problem may be considered as a variant of the Vehicle Routing Problem (VRP). Moreover, each delivery is governed by strict time windows, as

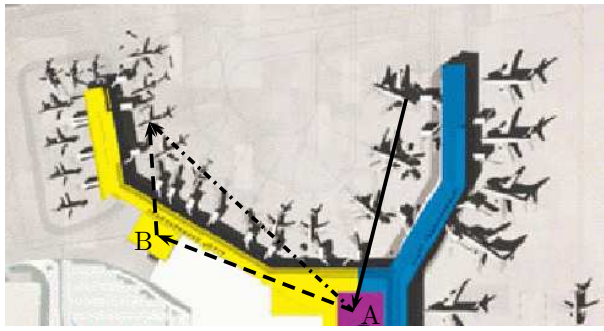


Figure C.1: Illustration of the considered problem. Bags are brought from the arriving airplanes to the baggage dispatch facility (A). From the dispatch facility (A) the bags can either be brought directly to the airplane (dotted line) or to the handling station (B), from where they are later brought to the airplane (dotted line).

all bags for a flight is required to be present a certain amount of time before take-off. A separate time window exists for the deliveries to the regular baggage handling stations. This problem is normally referred to as the Vehicle Routing Problem with Time Windows (VRPTW).

The VRP problem and its variants have been studied extensively over the years, see e.g. the comprehensive book by Toth and Vigo [21]. Most exact algorithms for VRP and VRPTW are based on column generation and branch-cut-price, see the surveyed by [3] and [13]. Recently the BCP framework was extended to include valid inequalities for the master problem, more specifically by applying the subset-row (SR) inequalities to the Set Partitioning master problem in [11] and later by applying Chvátal-Gomory Rank-1 (CG1) inequalities in [15]. Using an approach where columns with potentially negative reduced cost are enumerated (after good upper and lower bounds are found), [2] improved the lower bound by adding strengthened capacity inequalities and clique inequalities.

Apart from the capacity of the vehicles and the time windows at the nodes, the STBRP problem has a number of additional constraints. This includes the possibility of delivering a bag to one of two locations each having a different time window, the planning of multiple trips for each vehicle, and the possibility of splitting bags to the same flight. Some of the constraints have been studied before in the literature, but to the best of our knowledge not treated simultaneously.

The possibility of making a delivery to one of a number of locations, is normally referred to as the Generalized Vehicle Routing Problem (GVRP). This problem

has been studied by only a few references, including Ghiana and Improta [10].

The property of using a vehicle for repeated trips is referred to as the Vehicle Routing Problem with multiple trips (VRPM), which has been addressed in e.g. [5, 17].

The property of dispatching vehicles from more than one facility is usually referred to as the Multi Depot VRP (MDVRP), as presented in [14]. Tabu search heuristics for the MDVRP have been proposed by Renaud et al. [18] and Cordeau et al. [6].

The dispatcher may chose not to deliver all bags to a flight with the same vehicles. This may be a necessity if the number of bags for the flight exceeds the vehicle capacity. This is referred to as the Split Delivery VRP (SDVRP).

A unified framework for a large class of Vehicle Routing Problems with various extra constraints was presented in [20, 16] together with an adaptive large neighborhood search heuristic [19] for solving the problems.

C.1.2 Dynamic Problem

In an airport, it is often more relevant to consider the *online* or *dynamic* version of the problem, since we do not know the arrival times of airplanes for certain. The bags will hence arrive continuously as the inbound flights land and the bags marked for transfers are delivered to the dispatch facilities.

Most dynamic VRP problems studied in the literature assume that it is possible to modify a route on the fly when a new request enters the system. This is only possible if there is a method of detecting the present location of a vehicle and communicating a new route to the vehicle, e.g. by GPS and radio. We do not have this option here, so we only assign new routes to vehicles when they return to the facility. Moreover, when planning a new route for a vehicle, we only consider bags available at that time. We make no assumptions on the arrival times or destinations of bags arriving after that time.

Berbeglia [4] consider the dynamic pickup and delivery problem. Ankerl and Hammerle [1] presented an ant colony optimization algorithms for the dynamic pickup and delivery problem. Kozlak [12] consider a multi-agent approach for the online dynamic pickup and delivery problem. Gendreau et al. [9] consider a Dynamic Vehicle Dispatching Problem with Pick-ups and Deliveries.

For a theoretical discussion of online algorithms see Fiat and Woegninger [8].

C.1.3 Overview

The remaining paper is structured as follows: Section C.2 presents an IP-model of the static problem, assuming that all data are known in advance. Section C.3 describes the problem as a case study from a real-life baggage handling company and introduced uncertainty and the dynamic version of the problem. A weighted greedy algorithm for the dymamic problem is presented in Section C.4 and computational results are presented in Section C.5. Finally, conclusions and further work are discussed in Section C.6.

C.2 Formal Problem Description

The problem consists of N bags that must be delivered by K identical vehicles each having capacity Q. Each vehicle is allowed to return to the depot to pick up additional bags. For modeling purposes, a vehicle is allowed a maximum of R routes although R should be high enough to be unrestrictive. Each bag may be delivered in one of two locations (*flight* or *handling station*), each with its own time window. We model the problem as a graph problem where the flight and station of each bag are represented by separate nodes. For bag i, we represent the flight as node i and the handling station as node $N+i$. Each route begins at the starting depot (node 0) and ends at the ending depot (node $2N+1$). Thus, there are $2N+2$ nodes in the graph. We assume that all nodes are reachable from all other nodes, so the graph is complete. We allow a vehicle to have an empty route by simply driving directly from the start depot to the end depot.

The time windows of node i is given as $[a_i, b_i]$. The dropoff time needed at node i is given by s_i. The travel time between two nodes i and j in the graph is given by t_{ij}. The travel time between bags for the same location is set to 0. The arrival time of bag i is given by u_i.

The binary decision variable x_{ijkr} indicates if vehicle k drives from i to j on its r'th route. The binary variable z_i is set to one if bag i is not delivered. Finally, the continuous variable S_{ikr} is the time stamp when vehicle k arrives at node i on route r.

The resulting problem is defined on the graph $G = (V, E)$, where $V = \{0, \ldots, 2n+1\}$ is the set of nodes, and E is the set of edges. Let $\mathcal{N} = \{1, \ldots, n\}$ be the set of bags, $\mathcal{K} = \{1, \ldots, K\}$ be the set of vehicles, and $\mathcal{R} = \{1, \ldots, R\}$ the set of routes.

$$\min \sum_{i \in \mathcal{N}} z_i \tag{C.1}$$

$$\text{s.t.} \sum_{j \in V} \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}} x_{ijkr} + \sum_{j \in V} \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}} x_{n+i,j,k,r} + z_i = 1 \qquad \forall i \in \mathcal{N} \tag{C.2}$$

$$\sum_{j \in V} x_{0jkr} = 1 \qquad \forall k \in \mathcal{K}, r \in \mathcal{R} \tag{C.3}$$

$$\sum_{i \in V} x_{i,2n+1,k,r} = 1 \qquad \forall k \in \mathcal{K}, r \in \mathcal{R} \tag{C.4}$$

$$\sum_{\substack{(i,j) \in E \\ j \neq 2N+1}} x_{ijkr} \leq Q \qquad \forall k \in \mathcal{K}, r \in \mathcal{R} \tag{C.5}$$

$$\sum_{j \in V} x_{ijkr} - \sum_{j \in V} x_{jikr} = 0 \qquad \forall i \in V, k \in \mathcal{K}, r \in \mathcal{R} \tag{C.6}$$

$$x_{ijkr} = 1 \Rightarrow S_{ikr} + s_i + t_{ij} \leq S_{jkr} \qquad \forall (i,j) \in E, k \in \mathcal{K}, r \in \mathcal{R} \tag{C.7}$$

$$a_i \leq S_{ikr} \leq b_i \qquad \forall i \in V, k \in \mathcal{K}, r \in \mathcal{R} \tag{C.8}$$

$$S_{2n+1,k,r} \leq S_{0,k,r+1} \qquad \forall k \in \mathcal{K}, r \in \mathcal{R} \tag{C.9}$$

$$\sum_{j \in V} x_{jikr} = 1 \Rightarrow S_{0,k,r} \geq u_i \qquad \forall i \in V, k \in \mathcal{K}, r \in \mathcal{R} \tag{C.10}$$

$$x_{ijkr} \in \{0,1\} \qquad \forall (i,j) \in E, k \in \mathcal{K}, r \in \mathcal{R} \tag{C.11}$$

$$z_i \in \{0,1\} \qquad \forall i \in \mathcal{N} \tag{C.12}$$

$$S_{ikr} \geq 0 \qquad \forall i \in V, k \in \mathcal{K}, r \in \mathcal{R} \tag{C.13}$$

The objective (C.1) minimizes the number of undelivered bags. Constraint (C.2) sets $z_i = 1$ if bag i is not delivered to its flight or station on time. Constraints (C.3) and (C.4) ensure that each route leaves the depot once and returns to it once.

Constraint (C.5) ensures that the capacity of each vehicle is satisfied on all routes. Constraint (C.6) is a flow conservation constraint. Constraint (C.7) ensures that if edge (i, j) is used by vehicle k on route r, then the time stamp of node j is greater than the departure time at node i plus the travel time and dropoff time. The next constraint (C.8) maintains the time windows. Constraint (C.9) states for all vehicles and routes that route $r+1$ may not be started before route r has ended. Constraint (C.10) ensures that route r for vehicle k does not start before all the route's bags has arrived. If the vehicles are not identical, we can replace the righthand side in (C.5) with Q_k for vehicle k.

The problem is obviously $\mathcal{NP}$-hard since it contains the Hamilton cycle problem (defined on the flights) as special case when all time windows for the flights are open, time windows for the stations are closed, and only one vehicle is available for only one route.

Figure C.2 shows an example delivery with two dispatched routes.

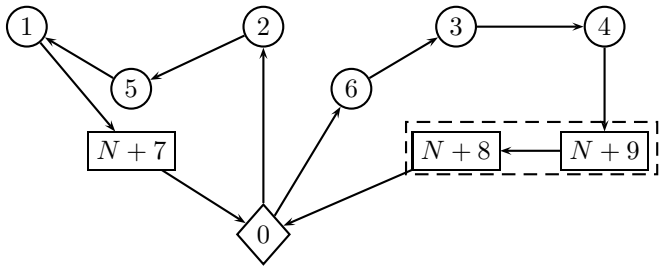


Figure C.2: Example delivery of 9 bags using two routes from the dispatch facility (diamond) to flights (round) and handling stations (square). Node numbers are shown in the node. Different nodes may correspond to the same physical location, as shown by the dashed box around nodes $N + 8$ and $N + 9$

To get more variance in the objective function, one may add a second objective to the problem to minimize the overall travel time

$$\min \sum_{(i,j)\in E} \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}} t_{ij} x_{ijkr}$$

This also stimulates that bags are brought to the handling stations, as the travel times to the stations are shorter than to the flights.

It is also natural to consider a weighted sum of the two objectives, since late deliveries of bags impose a cost which can be added to the travel costs.

C.3 Real-life case study

We consider the STBRP as a case study from a major European airport. Available data contains transfer bag data for one week of operations for each of the airport's two dispatch facilities.

The two dispatch facilities run independently and are placed in opposite areas of the airport. The northern facility, `N` covers arriving transfers from the northern part of the airport and the south facility `S` covers the southern part.

The facilities handle bags for *short transfers*, where the time between arrival and departure is relatively short. In particular, the baggage handling process for normal passengers, such as check-in, is in progress. This means that the transfer bags can be injected into the normal process. A bag can be injected into the normal process by delivering it to its handling station, if delivery to the aircraft has not yet begun. This normally happens 25 minutes before departure, which defines the end of the handling station's time window. As the process is operational for all bags, the start of the station time window is set such that a bag can never arrive early. The time window of the flight opens when the station closes and closes 5 minutes before departure, at which time the aircraft bay doors are closed. Unless a bag is too late (and thus undelivered), there is always at least one open time window open for the bag.

A period of time is required from the flight's arrival until it is available in the dispatch hall. This time includes de-loading at the aircraft, transportation to one of the central facilities and sorting and security screening if needed. Bags available for immediate dispatch are sent to the dispacth hall. Other transfer bags are either delivered to other airport facilities (such as storage areas), or set aside during the sorting step.

Each bag has a pickup area within the hall and a dedicated handling station. These locations are predetemined by the airport management to handle congestion issues within the airport.

The dispatch hall has a central driveway with 19 pickup areas on either side. To minimize traffic across the driveway, there are some movement restrictions within the hall. Vehicles must visit the pickup areas in forward order from entry to exit and can never drive backwards. Each vehicle has a turn radius that excludes the pickup area directly opposite the current area, as well as two areas on either side of the opposite area. Furthermore, a vehicle may only cross the driveway a maximum of two times to avoid congestion.

Some bags arrive very close to the scheduled departure time of the destination

flight. Such bags may be difficult or even impossible to deliver in time. How such bags are handled often depends on other circumstances: The destination flight may be delayed to wait for the bags (and passengers), or the passengers may be re-booked for a later departure. We denote such bags *rush bags* and assume that they are handled by a special, separate process that we do not consider. This might be a manual process within the dispatch hall, or a process handled by the airline itself. A bag is marked *rush* if it cannot be delivered to the flight in time.

The case company runs a fleet of identical vehicles with a capacity of 20 bags. A separate fleet is used for each dispatch facility. Facility `N` handles approximately 4000 short transfer bags every day and `S` handles approximately 7000. The handling of outbound bags is distributed over 7 baggage handling stations that each serve departing flights in a region of the airport. Handling stations handle all outgoing bags for both transfers and originating passengers and provides the simplest way to integrate transfer bags into the normal baggage handling process.

In addition to the available data, the case company has provided dispatch logs from the days of operation, detailing the delivery status of each bag, and its pickup time in the dispatch hall. The logs indicate that the company operates roughly 40 vehicles at dispatch hall `N` and 45 vehicles at `S`. On average, the company had 229 undelivered bags and 128 rush bags for `N`. For `S`, the company had 241 undelivered bags and 128 rush bags. For `N`, 50% of the delivered bags were delivered directly to the flight, and 62% of the delivered bags were delivered directly to the flight for `S`.

C.3.1 Problem Instances

To get a more realistic picture of the baggage dispatch operation, we consider the bag delivery times and the vehicle travel times as stochastic variables. We generate a series of scenarios with travel times generated from random distributions based on the original values.

For the vehicle travel times, we assume that each travel time t'_{ij} follows a normal distribution with mean value t_{ij} and standard deviation $0.1 \cdot t_{ij}$. This means that the uncertainty in the travel times increases with the expected travel time. This models a moderate level of uncertainty in vehicle operations that have neither very short or very long travel times. Excessive delays, such as taxiway crossings, would require a probability distribution with greater variance, for example the log-normal distribution used for arrival times described in the following.

For the bag arrival times, we assume that the average delivery time from the landed aircraft to the dispatch hall is 30 minutes, and that the delivery time follows a log-normal distribution. We use several distributions with mean value 30 minutes and different shape factors. Each shape factor is the base of a set of scenarios that describes an increasing amount of uncertainty in aircraft arrival times and bag delivery.

We create a series of randomly generated scenarios, where the random values are generated using the selected distributions. Each generated scenario set contains 20 randomly generated problem instances for each day of operation, giving 140 in total per set. We generate 4 sets per dispatch facility with increasing levels of uncertainty in bag arrival times, as modeled by using shape factors 0, 0.1, 0.2, and 0.3. All scenarios use normally distributed travel times as described above.

We name the scenario sets `N.x` and `S.x` for scenarios with normally distributed travel times generated with shape factor $0.1 \cdot x$. Thus, `N.1` will be generated using a scale factor of 0.1. Note that scale factor 0 imply that all bag delivery times take on the mean value, i.e. that there is no uncertainty. The increasing levels of uncertainty in bag arrival can model various degrees of flight delays, such as minor delays, airspace congestion or weather conditions.

Table C.1 provides an overview of the generated problem instances.

Scen.	$\|B\|$	B_R	B_s	t_f	t_s	B_I
`N.0`	3764.43	31.86	3060.00	5.10	3.25	36.63
`N.1`	3754.17	42.11	3074.41	5.09	3.24	37.14
`N.2`	3708.13	88.16	3062.90	5.02	3.20	37.21
`N.3`	3651.64	144.65	3043.44	4.95	3.16	37.37
`S.0`	6735.43	199.14	5496.15	6.51	3.22	33.90
`S.1`	6750.91	183.66	5499.50	6.54	3.23	34.42
`S.2`	6672.75	261.82	5475.19	6.48	3.19	34.52
`S.3`	6566.06	368.51	5442.50	6.38	3.14	34.72

Table C.1: Scenario data for generated scenarios. All values are averages of the 20 scenarios in each set. $|B|$ is the number of bags to schedule and B_R is the number of rush bags. B_s are the bags that can be delivered to the bag's handling station. t_f and t_s are the average travel times to flights and handling stations, respectively. B_I is the average time a bag can wait at the facility before it must be picked up.

C.4 Vehicle Dispatching

The infrastructure of the airport and dispatch facilities provide a series of restrictions on the possibilities for introducing computer-based support for scheduling and dispatching vehicles. The available information on arriving bags are limited to an information feed delivered by the airport. This feed reflects the last known position of each expected bag and is updated at certain points during the bag's journey. As the feed may be updated infrequently and irregularly, it can be difficult to track a bag on its way towards the dispatch hall. To avoid dispatching vehicles on erroneous assumptions, it is decided to only consider bags available for pickup in the hall at the time of dispatch.

Each vehicle operates with a level of uncertainty in the delivery times, which may be caused by several factors, for instance congestion at airport driveways or delivery points. Currently, the vehicles are not equipped with GPS or other means of electronic communication. There is therefore no easy way of establishing a vehicles expected return.

Due to these considerations, the presented algorithm is based on the following assumptions. All vehicles are scheduled individually and only for one trip at a time. The scheduling is done when a vehicle enters the dispatch hall, either when starting operation or when returning from a previous trip. Upon entering, a *route* is generated for the driver, which specifies a set of bags the driver should pick up in the hall and deliver throughout the airport. The route details the pickup location for each bag within the hall, and the delivery destinations for each bag.

The route is generated while the vehicle waits at the entry to the dispatch hall, so the computation times must be very fast. As there are many active vehicles that may return simultaneously, long calculations times would cause congestion to occur within the hall.

The dynamic aspect of the problem means that only a limited amount of the data can be considered when making decisions. The role of the algorithm is therefore to determine what constitutes a "good" individual route. There can be many ways to quantify a good route, of which most are conflicting. Examples include:

- Flights with an imminent departure time should be handled first. This reduces the risk of leaving a bag until it can no longer be delivered.
- The route should deliver as many bags as possible. This leaves fewer bags to handle for the vehicles arriving later and reduces the risk of creating

an unsurmountable amount of bags.

- Routes should be as short as possible. Shorter routes increase the number of routes which can be managed with the given amount of vehicles. This is especially important for delivering bags with a tight time window.

C.4.1 The Algorithm

To satisfy the above requirements, we present a parameterized greedy algorithm with linear penalty functions. The algorithm is run each time a vehicle enters the dispatch hall, and calculates a set of bags for the next route. The algorithm considers all bags $B \subseteq \mathcal{N}$ present in the dispatch hall at the time of calculation. We define the subgraph $G[B] = (V[B], E[B]) \subseteq G$ as the induced subgraph containing only nodes and edges belonging to the depot or the bags in B. To each edge incident with the depot, we assign a cost that reflects the attractiveness of delivering the bag associated with that node. Edges with low cost describe the bags that are most attractive to deliver. To compute the cost of using edge (i, j) we use the function $\sigma_{ij}(S_{ikr})$, where S_{ikr} is the time where service at node i is completed. At the beginning of the algorithm, S_{0kr} is the time the vehicle enters the dispatch hall.

We compute the cost of all edges incident to the depot as

$$c(e_{0j}) = \sigma_{0j}(S_{0kr}).$$

The edge with lowest cost (and thereby the associated bag) is selected as the beginning of the route, which we denote r_1.

For subsequent bags, the edges e_{ij} incident to the last delivery point i is penalized again using a more complex penalty function:

$$c(e_{ij}) = \begin{cases} \sigma_{ij}(S_{ikr}), & j \leq 2n, a_j \leq S_{ikr} + t_{ij} \leq b_j \\ \sigma_T \cdot \sigma_{0r_1}(S_{0kr}), & j = 2n+1 \\ \infty & \text{hall restrictions or time window violated} \end{cases} \tag{C.14}$$

The node with minimum cost edge $j' = \arg\min_j c(e_{ij})$ is selected as the next stop on the route. $S_{j'kr}$ is then updated to reflect the new availability time of the vehicle $S_{j'kr} = S_{ikr} + t_{ij'} + s_{j'}$.

The edges incident to j' are then penalized again using equation (C.14).

When the cheapest edge is the edge that returns to the depot, the route ends. The cost of the return edge is fixed for all non-depot nodes as a factor σ_T

times the cost of the first edge. We denote σ_T the *termination parameter*, as it determines the maximum cost of any bag to add to the route. The termination is set relative to the first bag to ensure a *rush-like* behavior. If the first bag has a very low cost, it is likely a near-critical bag. The route should then be less likely to include other bags that must be picked up or delivered. If, on the other hand, the initial bag has a high penalty, is is not likely to be critical, and the driver will have more slack to include other stops on the route.

The cost of an edge e_{ij} is ∞ if the bag cannot be picked up (due to hall movement restrictions), or if the node j is visited outsides its time window (i.e. $S_{ikr}+t_{ij} < a_j$ or $S_{ikr}+t_{ij} > b_j$). During the generation of a route, the set of nodes that can be reached using edges with finite cost can change. As the vehicle's time increases, the windows of some nodes will close as the respective station or flight closes. The flight node is only available after the time window at the corresponding station is closed. This means that some nodes may also become available as S_{jkr} is increased. The link between the time windows at the station and the flight means that a destination is always available for the delivery an available bag. For this reason, the normal VRPTW practice of allowing early vehicles to wait at the customer is not applied here.

What we solve is essentially the dynamic prize-collecting TSP on a restricted graph. Feillet et al. [7] give an excellent overview of exact and heuristic algorithms for the TSP with profits. The problem may be seen as a bicriteria TSP with two opposite objectives: maximize profit and minimize travel distance. Most researchers address the problem in single-objective version where the two objectives are weighted and aggregated. In this paper we use a weighted sum of three objectives, which use dynamic, time-dependent, edge costs. The edge cost function σ is described in detail in the following section.

C.4.2 Penalty Function

The edge cost $c(e_{ij})$ of traveling from i to j, as defined by (C.14), is calculated in part by the time-dependent penalty function $\sigma_{ij}(S_{ikr})$. Visiting node j is equivalent to delivering bag $(j \bmod N)$ to either its flight (if $j \leq N$) or its handling station (if $j > N$). The cost of adding this bag is calculated as a weighted sum of three terms, that each take different aspects of the delivery or the route into account. The cost function $\sigma_{ij}(S_{ikr})$ is defined as

$$\sigma_{ij}(S_{ikr}) = \alpha_L \cdot \sigma_L(i,j) + \alpha_R \cdot \sigma_R(S_{ikr}+t_{ij}-S_{0kr}) + \alpha_D \cdot \sigma_D(S_{ikr}+t_{ij}-b_{j \bmod N}). \tag{C.15}$$

The cost function (C.15) is a weighted sum of three terms with real-valued weights α_L (location), α_R (route length) and α_D (departure) that are parame-

ters to the algorithm. Each term is an individual penalty function that evaluates a specific part of the route that would result in adding the bag. As the functions consider different aspects of the route, they also take different parts of the existing route as parameters.

Each measure introduces a level of robustness to the selection, as they in different ways penalize bags that are not critical or may be better delivered as part of another route.

The *location* penalty function $\sigma_L(i, j)$ penalizes a change in location, depending on the location type. It is defined as a discrete value for each type of location: Delivery at flight for $i \leq N$ and delivery at handling station for $i > N$. The discrete values of $\sigma_L(i, j)$ are defined as

$$\sigma_L(i,j) = \begin{cases} 1 & i \leq N, j \leq N \\ 10 & i > N, j \leq N \\ 0 & i \leq N, j > N \\ 1 & i > N, j > N \end{cases} \tag{C.16}$$

No penalty is incurred for changing from a flight to a handling station, which encourages additional bags to be brought along with a delivery to a flight, as long as the subsequent bags are more robust deliveries to a handling station. Added bags for a flight after visiting a station incurs a penalty of 10, making it practically illegal. No penalty is incurred if the location is not changed, i.e. several bags are brought to the same location.

The two remaining penalty terms in equation (C.15) are calculated using piecewise linear penalty functions of their parameter value x. By using this approach, we can modify the behavior of the penalty to fit desired properties of the route.

The *route length* penalty function $\sigma_R(x) = \sigma_R(S_{ikr} + t_{ij} - S_{0kr})$ penalizes the total expected route length in minutes. When considering this function isolated, routes are preferred to be short. Short routes minimizes uncertainty and maximizes the flow of vehicles at the dispatch hall.

The *departure* penalty function $\sigma_D(x) = \sigma_R(S_{ikr} + t_{ij} - b_{j \bmod N})$ penalizes the difference between the expected delivery of the bag, and the flights departure, modeled by the end of the time window of the bag's flight node $j \bmod N$. This function defines the cost of the bag in terms of the flight's departure.

We consider different penalty functions for each of the two terms. The considered functions are depicted in Figure C.3. Functions A and B are general functions that are considered for both terms. Function A has a constant value

of 0.5 and B increases linearly from 0 to 1. Function C is considered only for route length penalty σ_R and has a small increasing penalty for routes up to 30 minutes in length. The penalty then increases steeply for routes longer than 30 minutes. This function puts more emphasis on shorter routes than B, which should increase the frequency of vehicles returning to the dispatch center.

Functions D and E are considered only for departure penalty σ_D. The penalty for both functions is highest in the interval from 25 to 35 minutes, where the penalty is 1. The effect of this peak is to avoid deliveries very close to the closing time of the handling station, which closes at 25 minutes before departure. Arriving close to the station's closing has a high risk of being late, and hence being redirected directly to the flight.

The function D decreases from the peak towards a penalty of 0.4 at 0 and 50 minutes, giving a penalty function that is roughly symmetric around the peak. Function E combines aspects from B and D by having a penalty that increases from 0 at 0 minutes like B, but the same peak as D. Functions D and E penalizes deliveries that risk redirection from the station to the flight.

C.4.3 Simulation

The algorithm is intended to support dispatching during a day of operation, so the expected arrival times and travel times are not known in advance. To simulate this, the dispatching of a vehicle is done in two phases. First, the algorithm needs to select the bags to include in the route. In this phase, expected travel times are used. This means that the feasibility checks and bag cost penalty are calculated using the expected (mean) travel times. When a route is finished, the second phase determines the actual completion time of the route. The route is traversed using the actual (randomly generated) travel times. If the vehicle arrives too late at a flight, the bag is considered *undelivered*. If the vehicle arrives too late at a handling station, the vehicle is *redirected* to the target flight. The flight location is inserted as an extra delivery point, before any subsequent deliveries. Redirects extend the duration of the route, and increase the risk of subsequent redirects or undelivered bags. In case of severe delays, it is possible for a bag to be both redirected and refused.

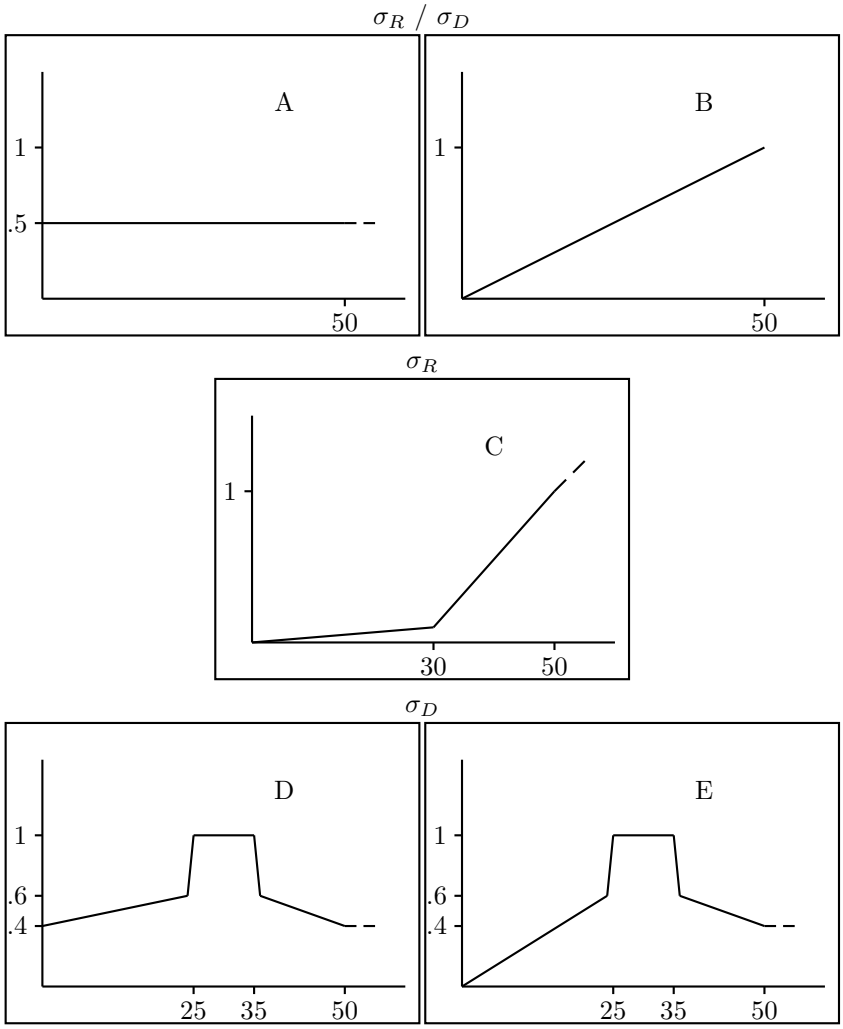


Figure C.3: Bag penalty profile for σ_R (A,B,C) and σ_D (A,B,D,E). Each profile shows the value for the base value of the penalty function, total route length (σ_R) or remaining time after delivery (σ_D).

C.5 Computational Results

We present the results of experiments run on the generated instances presented in Table C.1 of Section C.3.1. The algorithm was implemented using C# .NET. The instances were generated using probability distributions implemented in the `Troschuetz.Random` software library [22]. All instances were solved in less than one second, which makes the algorithm applicable in the dynamic dispatching scenario.

We structure the experiments as follows. In Section C.5.1, we consider the parameters and penalty functions on a fixed number of vehicles. In Section C.5.2 we investigate the robustness of the algorithm on various fleet sizes.

C.5.1 Weight Parameters

We evaluate different combinations of weights $\alpha_L, \alpha_R, \alpha_D$ and different termination threshold values $\sigma_T = \{0, 1, 2, 3, 4\}$. Values for $(\alpha_L, \alpha_R, \alpha_D)$ are chosen as sample coordinates of the unit simplex, such that $\alpha_L + \alpha_R + \alpha_D = 1$.

For the tuning we use a sample fleet size of 25 vehicles on all days for the instances in set `S.1`. Not all bags can be delivered with this fleet size, so the tuning also illustrates the performance of different parameters under a degree of pressure.

To also consider the effect on the parameters when using different penalty functions, we present results the full combination of parameters for three sample functions $(\sigma_D, \sigma_R) = \{(B, C), (D, C), (E, C)\}$. See Figure C.3 and Section C.4.2 for a definition of the penalty functions.

The effect of the three penalty function combinations is presented in three separate tables. Table C.2 presents the results for $(\sigma_D, \sigma_R) = (B, C)$. Combinations $(\sigma_D, \sigma_R) = (D, C)$ and $(\sigma_D, \sigma_R) = (E, C)$ are presented in tables C.3 and C.4, respectively.

Each table show the average number of undelivered bags for each combination of weights and termination threshold. The rightmost column reports the average across all termination penalizes for each parameter combination and the last row shows the average for each termination threshold.

The three tables show similar results. Looking at the average number of undelivered bags per parameter combination, the same combinations score well in all

			σ_T					
α_L	α_R	α_D	0	1	2	3	4	
0.00	0.00	1.00	580.19	342.38	236.57	256.51	264.87	336.10
0.00	0.20	0.80	580.19	580.19	107.74	117.58	117.54	300.65
0.00	0.40	0.60	580.19	580.19	100.25	103.94	104.38	293.79
0.00	0.50	0.50	580.19	580.19	94.62	96.11	96.66	289.55
0.00	0.60	0.40	580.19	580.19	89.07	93.35	91.53	286.86
0.00	0.80	0.20	580.19	580.19	108.72	75.65	74.49	283.85
0.00	1.00	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
0.10	0.10	0.80	580.19	580.19	111.79	117.03	120.60	301.96
0.10	0.40	0.50	580.19	580.19	99.86	99.21	102.05	292.30
0.10	0.50	0.40	580.19	580.19	93.24	96.11	92.11	288.37
0.10	0.80	0.10	580.19	580.19	265.02	76.88	**62.36**	312.93
0.20	0.00	0.80	580.19	342.38	236.57	256.51	264.87	336.10
0.20	0.20	0.60	580.19	580.19	108.51	109.14	115.59	298.72
0.20	0.40	0.40	580.19	580.19	94.69	96.51	97.41	289.80
0.20	0.60	0.20	580.19	580.19	106.49	80.91	85.41	286.64
0.20	0.80	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
0.25	0.25	0.50	580.19	580.19	101.80	109.55	106.64	295.67
0.25	0.50	0.25	580.19	580.19	93.93	87.38	89.76	286.29
0.33	0.33	0.33	580.19	580.19	94.56	96.55	96.53	289.60
0.40	0.00	0.60	580.19	342.38	236.57	254.15	264.87	335.63
0.40	0.10	0.50	580.19	580.19	111.84	116.34	118.69	301.45
0.40	0.20	0.40	580.19	580.19	102.14	109.51	106.54	295.71
0.40	0.40	0.20	580.19	580.19	92.89	87.81	89.58	286.13
0.40	0.50	0.10	580.19	580.19	149.13	72.71	69.56	290.36
0.40	0.60	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
0.50	0.00	0.50	580.19	342.38	236.57	256.51	264.87	336.10
0.50	0.10	0.40	580.19	580.19	107.74	117.58	117.54	300.65
0.50	0.25	0.25	580.19	580.19	94.62	96.11	96.66	289.55
0.50	0.40	0.10	580.19	580.19	108.72	75.65	74.49	283.85
0.50	0.50	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
0.60	0.00	0.40	580.19	342.38	236.57	256.51	264.87	336.10
0.60	0.20	0.20	580.19	580.19	94.69	96.51	97.41	289.80
0.60	0.40	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
0.80	0.00	0.20	580.19	342.38	236.57	256.51	264.87	336.10
0.80	0.10	0.10	580.19	580.19	94.69	96.51	97.41	289.80
0.80	0.20	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
1.00	0.00	0.00	1327.16	1327.16	1327.16	1327.16	1327.16	1327.16
			745.34	706.78	384.28	379.29	380.60	519.26

Table C.2: Average number of undelivered bags for parameter and termination weight combinations, using penalty functions $(\sigma_D, \sigma_R) = (B, C)$. Best performance is highlighted in bold. The rightmost column and bottom row show average values.

			σ_T					
α_L	α_R	α_D	0	1	2	3	4	
0.00	0.00	1.00	2026.12	2125.12	1532.59	1518.64	1518.63	1744.22
0.00	0.20	0.80	2026.12	2026.12	726.56	707.19	707.41	1238.68
0.00	0.40	0.60	2026.12	2026.12	685.11	658.47	655.14	1210.19
0.00	0.50	0.50	2026.12	2026.12	652.06	627.92	633.23	1193.09
0.00	0.60	0.40	2026.12	2026.12	590.56	588.77	587.05	1163.72
0.00	0.80	0.20	2026.12	2026.12	409.11	381.82	404.25	1049.48
0.00	1.00	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
0.10	0.10	0.80	2026.12	2026.12	738.79	721.57	721.57	1246.83
0.10	0.40	0.50	2026.12	2026.12	668.05	654.68	652.80	1205.55
0.10	0.50	0.40	2026.12	2026.12	622.46	608.26	608.91	1178.38
0.10	0.80	0.10	2026.12	2026.12	418.12	294.16	**285.89**	1010.08
0.20	0.00	0.80	2026.12	2125.12	1532.59	1518.64	1518.63	1744.22
0.20	0.20	0.60	2026.12	2026.12	712.37	700.24	699.84	1232.94
0.20	0.40	0.40	2026.12	2026.12	652.14	627.92	633.23	1193.11
0.20	0.60	0.20	2026.12	2026.12	462.26	448.14	453.86	1083.30
0.20	0.80	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
0.25	0.25	0.50	2026.12	2026.12	692.64	682.65	682.41	1221.99
0.25	0.50	0.25	2026.12	2026.12	553.21	539.55	541.12	1137.23
0.33	0.33	0.33	2026.12	2026.12	651.98	627.94	633.23	1193.08
0.40	0.00	0.60	2026.12	2125.12	1532.59	1518.64	1518.63	1744.22
0.40	0.10	0.50	2026.12	2026.12	727.74	712.68	712.72	1241.08
0.40	0.20	0.40	2026.12	2026.12	691.74	683.13	682.89	1222.00
0.40	0.40	0.20	2026.12	2026.12	553.21	539.43	541.15	1137.21
0.40	0.50	0.10	2026.12	2026.12	383.44	353.48	362.44	1030.32
0.40	0.60	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
0.50	0.00	0.50	2026.12	2125.12	1532.59	1518.64	1518.63	1744.22
0.50	0.10	0.40	2026.12	2026.12	726.56	707.19	707.41	1238.68
0.50	0.25	0.25	2026.12	2026.12	652.06	627.92	633.23	1193.09
0.50	0.40	0.10	2026.12	2026.12	409.11	381.82	404.25	1049.48
0.50	0.50	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
0.60	0.00	0.40	2026.12	2125.12	1532.59	1518.64	1518.63	1744.22
0.60	0.20	0.20	2026.12	2026.12	652.14	627.92	633.23	1193.11
0.60	0.40	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
0.80	0.00	0.20	2026.12	2125.12	1532.59	1518.64	1518.63	1744.22
0.80	0.10	0.10	2026.12	2026.12	652.14	627.92	633.23	1193.11
0.80	0.20	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
1.00	0.00	0.00	1327.16	1327.16	1327.16	1327.16	1327.16	1327.16
			1917.73	1933.78	920.31	903.10	905.25	1316.03

Table C.3: Average number of undelivered bags for parameter and termination weight combinations, using penalty functions $(\sigma_D, \sigma_E) = (D, C)$. Best performance is highlighted in bold. The rightmost column and bottom row show average values.

			σ_T					
α_L	α_R	α_D	0	1	2	3	4	
0.00	0.00	1.00	639.34	705.84	784.21	764.31	759.95	730.73
0.00	0.20	0.80	639.34	639.34	277.32	273.66	272.81	420.49
0.00	0.40	0.60	639.34	639.34	245.88	246.94	247.88	403.87
0.00	0.50	0.50	639.34	639.34	234.80	220.29	221.89	391.13
0.00	0.60	0.40	639.34	639.34	209.59	196.79	200.50	377.11
0.00	0.80	0.20	639.34	639.34	158.90	141.62	142.45	344.33
0.00	1.00	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
0.10	0.10	0.80	639.34	639.34	292.03	280.04	280.90	426.33
0.10	0.40	0.50	639.34	639.34	233.87	235.39	235.18	396.62
0.10	0.50	0.40	639.34	639.34	218.99	210.11	213.61	384.28
0.10	0.80	0.10	639.34	639.34	184.41	120.78	**112.49**	339.27
0.20	0.00	0.80	639.34	705.84	784.21	764.31	759.95	730.73
0.20	0.20	0.60	639.34	639.34	276.84	263.54	263.84	416.58
0.20	0.40	0.40	639.34	639.34	234.45	220.16	221.91	391.04
0.20	0.60	0.20	639.34	639.34	165.20	162.22	159.27	353.07
0.20	0.80	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
0.25	0.25	0.50	639.34	639.34	263.44	249.92	252.69	408.95
0.25	0.50	0.25	639.34	639.34	195.76	187.01	179.09	368.11
0.33	0.33	0.33	685.24	685.24	232.91	236.59	235.50	415.09
0.40	0.00	0.60	639.34	705.84	784.21	764.31	759.95	730.73
0.40	0.10	0.50	639.34	639.34	283.51	276.26	275.75	422.84
0.40	0.20	0.40	639.34	639.34	263.98	249.94	252.69	409.06
0.40	0.40	0.20	639.34	639.34	196.14	187.64	179.62	368.41
0.40	0.50	0.10	639.34	639.34	152.31	132.46	125.86	337.86
0.40	0.60	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
0.50	0.00	0.50	639.34	705.84	784.21	764.31	759.95	730.73
0.50	0.10	0.40	639.34	639.34	277.32	273.66	272.81	420.49
0.50	0.25	0.25	639.34	639.34	234.80	220.29	221.89	391.13
0.50	0.40	0.10	639.34	639.34	158.90	141.62	142.45	344.33
0.50	0.50	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
0.60	0.00	0.40	639.34	705.84	784.21	764.31	759.95	730.73
0.60	0.20	0.20	639.34	639.34	234.45	220.16	221.91	391.04
0.60	0.40	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
0.80	0.00	0.20	639.34	705.84	784.21	764.31	759.95	730.73
0.80	0.10	0.10	639.34	639.34	234.45	220.16	221.91	391.04
0.80	0.20	0.00	1474.17	1474.17	1474.17	1474.17	1474.17	1474.17
1.00	0.00	0.00	1327.16	1327.16	1327.16	1327.16	1327.16	1327.16
			794.54	805.33	549.67	538.52	537.48	645.11

Table C.4: Average number of undelivered bags for parameter and termination weight combinations, using penalty functions $(\sigma_D, \sigma_R) = (E, C)$. Best performance is highlighted in bold. The rightmost column and bottom row show average values.

three tables. The combinations $(\alpha_L, \alpha_R, \alpha_D) = (0.0, 0.8, 0.2)$ and $(\alpha_L, \alpha_R, \alpha_D) = (0.4, 0.5, 0.1)$ are among the three best in all tables. The combination $\alpha = (\alpha_L, \alpha_R, \alpha_D) = (0.1, 0.8, 0.1)$ has the best average in Table C.3 ($\sigma_D = D$) and Table C.4 ($\sigma_D = E$). In Table C.2 that summarizes $\sigma_D = B$, the average number for this combination is not among the best, but the best individual value across all tables in found in this table for this combination, with 62.36 undelivered bags on average.

It is interesting that while the best combinations favor the departure term α_D to be low, the combinations with $\alpha_D = 0$ consistently produces results significantly worse than all others. Combinations with $\alpha_R = 0$ are also consistently poor. This is perhaps less surprising, since α_R is high in the best combinations. The weight parameter that seems to be least significant is α_L. Several combinations with $\alpha_L = 0$ produce decent results, with $\alpha = (\alpha_L, \alpha_R, \alpha_D) = (0.0, 0.8, 0.2)$ being in the top three on average values. This is also not surprising, as the location change function is to some extent supported by the other functions as well: Keeping routes short and prioritizing critical bags will also minimize the number of visited locations and form *flight first* routes.

Considering the termination parameter σ_T, the values $\sigma_T = 3$ and $\sigma_T = 4$ have the best performances. In all three tables, the average distance between undelivered bags for $\sigma_T = 3$ and $\sigma_T = 4$ are small, less than 3. This also shows a good level of consistency across the tables.

The global average for each table shows 519 undelivered bags on average for $\alpha_D = B$ versus 1316 for $\alpha_D = D$ and 645 for $\alpha_D = E$. From this we conclude that $\alpha_D = B$ is the best choice for α_D.

In all three tables, the best individual performance is found using $\alpha = (\alpha_L, \alpha_R, \alpha_D) = (0.1, 0.8, 0.1)$ and $\sigma_T = 4$. These values are also shown in bold in the tables.

We use these parameters to evaluate all the σ_D and σ_R penalty functions, as presented in Section C.4.2, $\sigma_R = \{A, B, C\}$, and $\sigma_D = \{A, B, D, E\}$. The results of running each penalty function combination on `S.1` are presented in Table C.5.

As seen from the table, the results of using $\sigma_R = C$ is in all cases better than any results with $\sigma_R = A$ or $\sigma_R = B$. For each group of results with the same σ_R function, the best results are for $\sigma_D = B$. The observations are emphasized by the best overall result, which is found for the combination $(\sigma_D, \sigma_R) = (B, C)$ with 62 undelivered bags on average.

A direct connection is seen between the number of delivered bags, the route length and the number of bags per route. The top five results are the only combinations with average loads above 4 and average route lengths above 10

σ_D	σ_R	D	D_s	D_f	U_s	U_f	$\|R\|$	R_{len}	R_{load}
A	A	5276.74	3746.48	1530.26	0.00	1474.17	2041.67	8.52	3.30
B	A	6177.06	2466.70	3710.36	0.00	573.85	1775.08	9.78	3.80
D	A	4724.79	3258.44	1466.34	0.00	2026.12	2092.46	8.23	3.22
E	A	6111.57	3078.50	3033.07	0.00	639.34	1999.48	8.45	3.37
A	B	6242.80	4980.04	1262.76	0.00	508.11	1739.46	9.01	3.88
B	B	6382.81	3329.38	3053.44	0.00	368.09	1493.36	10.75	4.52
D	B	5121.41	3627.72	1493.69	0.00	1629.50	1981.01	8.46	3.40
E	B	6202.25	3438.75	2763.50	0.00	548.66	1909.26	8.66	3.53
A	C	6574.13	4888.99	1684.96	0.17	176.78	1216.73	12.10	5.54
B	C	6688.54	3945.13	2743.06	0.35	62.36	1086.56	14.01	6.21
D	C	6465.02	4380.49	2084.37	0.16	285.89	1160.41	12.81	5.81
E	C	6638.42	4320.98	2317.26	0.18	112.49	1143.71	12.94	5.90

Table C.5: Results of simulations on `S.1` using different combinations of penalty function profiles for σ_D and σ_R. D, D_s and D_f are the number of delivered bags in total, to the handling stations and to the flights, respectively. U_s and U_f are undelivered bags at stations (redirections) and at flights. $|R|$, R_{len} and R_{load} is the average number of routes, and the average length (in minutes) and vehicle load per route.

minutes. The best result has the highest average load and route duration.

The four best results are the only results with a significant number of redirections. This could indicate that they deliver bags to the handling stations near the end of the station time windows and thus risking redirections. It seems that higher risk willingness yields better results, as the result improve with the number of redirections.

In some cases, minimizing the number of redirections may be preferable. This may be to emphasize a risk aversion strategy, or to reduce unnecessary traffic in congested handling stations. We have chosen to consider the objective of delivered bags only, since we do not have realistic data to model other requirements. Also, the number of redirections is low in all cases, with a maximum of 0.35 on average. However, the results clearly show the algorithm's ability to introduce more risk aversion. The combination $(\sigma_D, \sigma_R) = (E, C)$ has 112 undelivered bags on average with 0.18 redirections, about half of the maximum.

C.5.2 Fleet Size and Uncertainty

In this section we consider the effect of varying the fleet size on all `N.x` and `S.x` scenarios. We use the parameters determined in the previous section: $(\alpha_L, \alpha_R, \alpha_D) = (0.1, 0.8, 0.1)$, $\sigma_T = 4$, and penalty functions $(\sigma_D, \sigma_R) = (B, C)$.

Each scenario is evaluated with varying fleet sizes from 20 to 45. Figure C.4 shows the average number of undelivered bags as a function of the fleet size for scenarios `S.0`, `S.1`, `S.2`, and `S.3`. For `S.0` the number of undelivered bags reaches a minimum for all fleet sizes larger than 26. For the remaining scenarios, the number of undelivered bags decrease towards this minimum for all fleet sizes, all reaching values close to the minimum at 45 vehicles.

The most difficult instances to solve are `S.1`, which have the largest number of undelivered bags for almost all fleet sizes. For `S.2` and `S.3`, there are fewer bags to deliver since more are marked *rush bags*. This is also seen in Table C.1.

Another view on the solutions to the `S.x` scenarios is provided in Figure C.5. The figure shows the number of instances with 0 undelivered bags for scenarios `S.0`, `S.1`, `S.2`, and `S.3`.

As seen from the figure, the number of completely solved instances stabilizes at around 45% at 28 vehicles for `S.0`. For the other `S.x`-scenarios, less than 10% are completely solved for 45 vehicles.

Figure C.6 shows undelivered bags for the `N.x` scenarios as function of the fleet size. These scenarios are smaller than the `S.x` scenarios, and the overall number of undelivered bags are similarly smaller. The `N.0` instances reaches a minimum at 25 vehicles and the other scenarios also have few undelivered bags. At 36 and more vehicles, the number of undelivered bags is reduced very little.

Figure C.7 displays the number of instances with no undelivered bags for scenarios `N.0`, `N.1`, `N.2`, and `N.3`. About 60% of the `N.0` can be completely dispatched when using 28 or more vehicles. For the other scenarios, the number of completely dispatched scenarios increases with the fleet size. About 20% – 25% of these instances are completely dispatched, with the `N.1` being the most difficult.

For all scenarios, a connection is seen between the number of vehicles and the number of undelivered bags. Compared to the delivery numbers of the customer case, we see a clear improvement when using the same number of vehicles. Both `N` and `S` had more than 200 undelivered bags on average. In tables C.4 and C.6 we see that all scenarios can be solved better for all presented fleet sizes. Based on the generated scenarios, the same level of service or better can be achieved

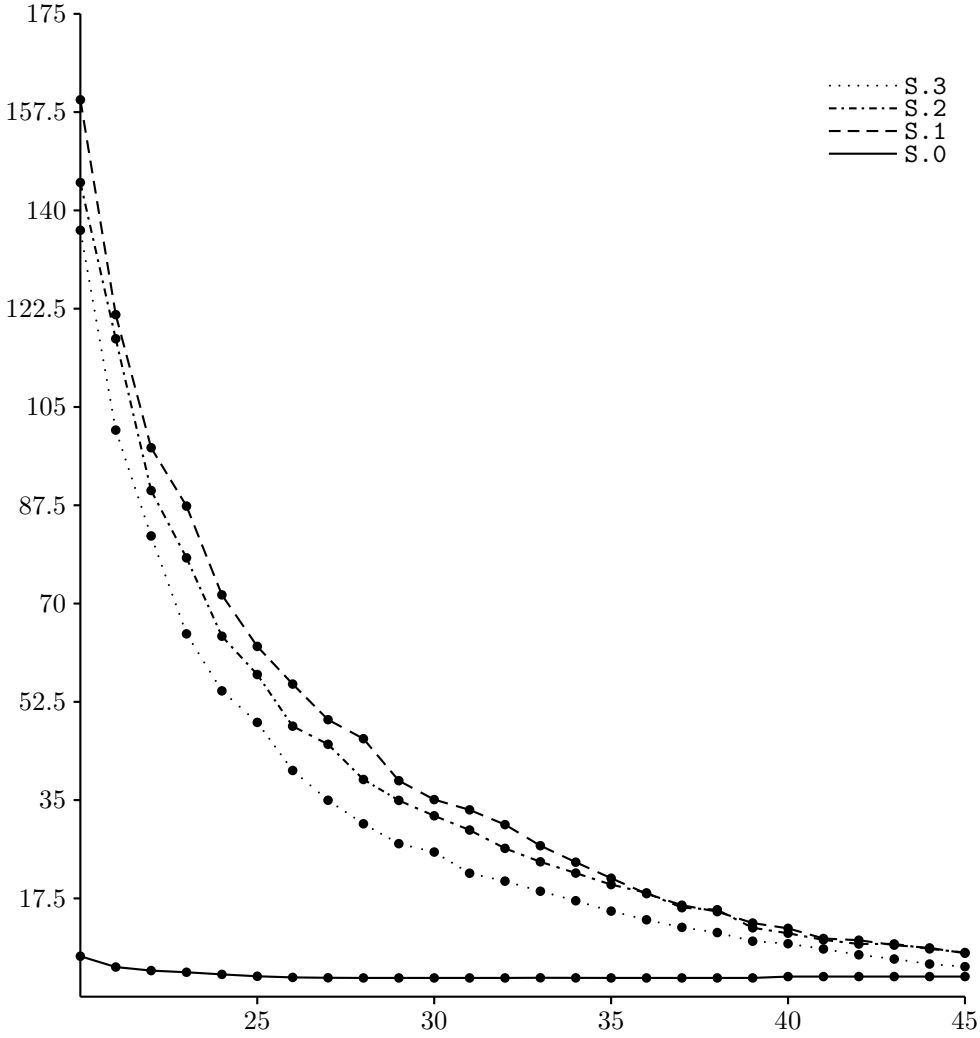


Figure C.4: Average undelivered bags per fleet size for dispatch facility S with various levels of uncertainty. The x-axis depicts fleet size and the y-axis is the number of undelivered bags.

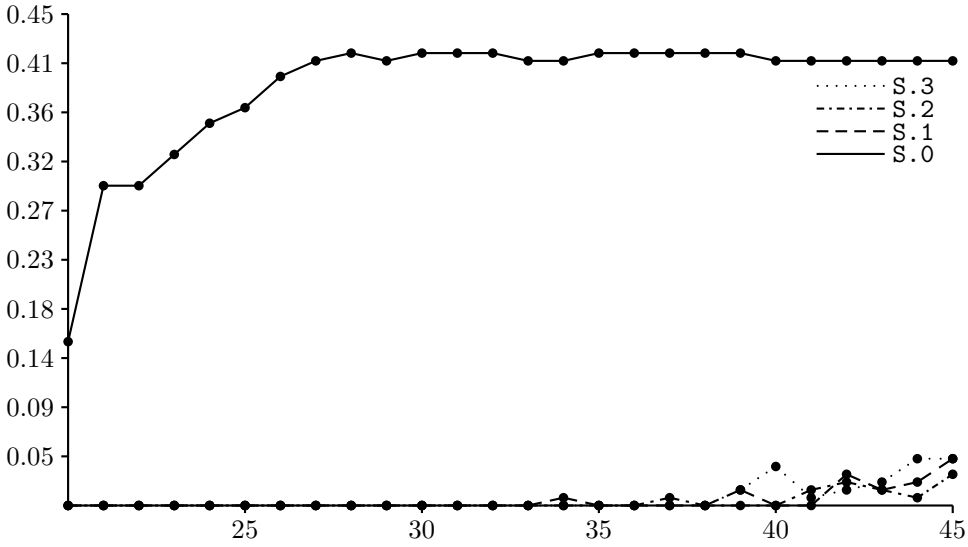


Figure C.5: Percentage of instances (y-axis) with no undelivered bags for `S.x`-scenarios for various fleet sizes (x-axis).

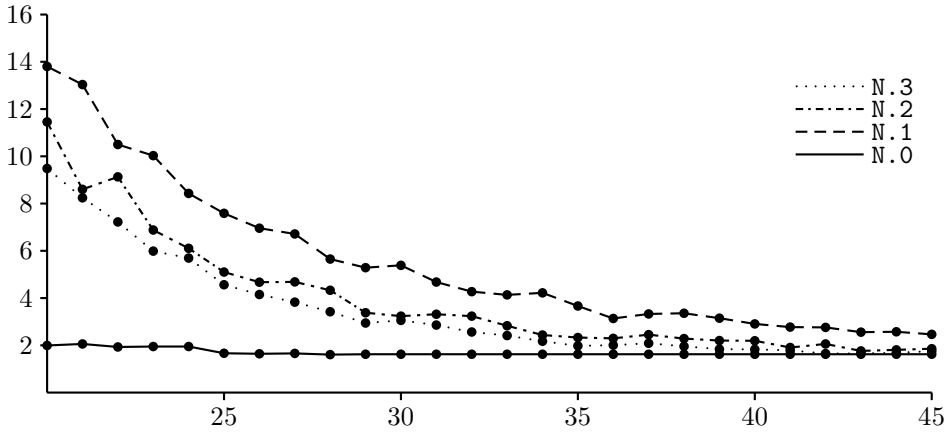


Figure C.6: Average undelivered bags per fleet size for dispatch facility `N` with various levels of uncertainty. The x-axis depicts fleet size and the y-axis is the number of undelivered bags.

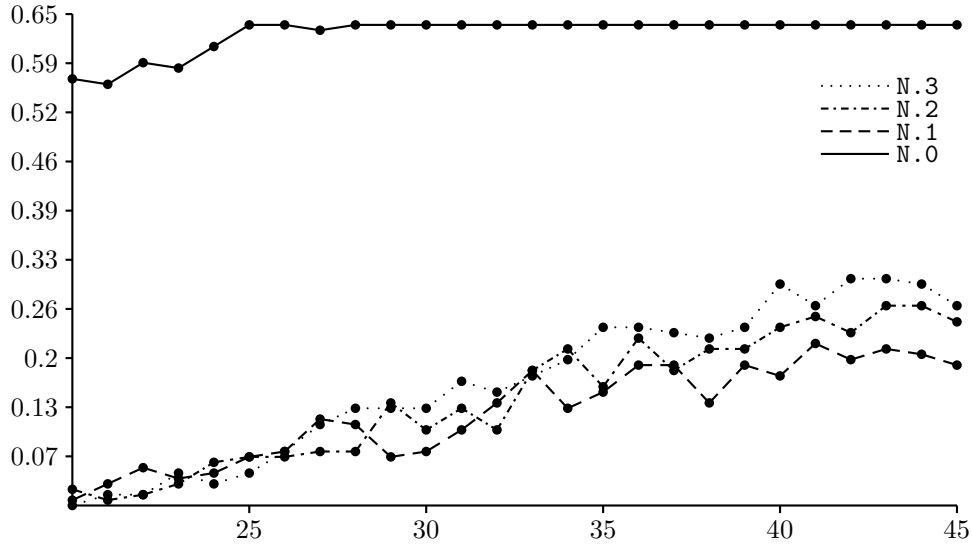


Figure C.7: Percentage of instances with no undelivered bags for `N.x` scenarios for various fleet sizes.

with less than half the current fleet size even when considering a substantial level of uncertainty.

Although it is clearly preferable to deliver all bags, it may be acceptable to leave a portion of the bags undelivered. This might be the case, if the service contract, as common, specifies a certain level of expected service. In this case, the company can use simulations like the one presented here to adjust the fleet size according to the contractual needs, thus saving money on expenditures. Furthermore, the simulations can determine the expected fleet size needed for achieving different service levels. This may be useful in contract negotiations.

C.6 Conclusion and Future Work

In this paper we have presented the Short Transfer Baggage Routing Problem from a real-life application in a major European airport. A fast parameterized greedy algorithm has been presented and evaluated on airport data. Several time-dependent penalty functions have been presented and evaluated. Experimental results show that a trade-off can be made between a maximum delivery

of bags and a risk aversion policy that reduces redirections and route extensions.

The algorithm has been evaluated further to show that the results are robust with regards to stochastic bag arrivals and vehicle transport times. The algorithm was also shown to function well under a shortage of vehicles, showing that vehicle breakdowns or reduced manning can be handled, or that savings can be made in fleet size at a cost of fewer delivered bags.

It is interesting to observe, that among several dispatching strategies it seems that risk willingness pays off, also when there is a large degree of uncertainty. This is due to the structure of the considered problem, where it seems to be attractive to schedule bags for the handling station even when the time limit is very tight. If a bag arrives later than the time window, the vehicle will have to bring the bag directly to the airplane. But if the vehicle does not try to deliver the bag at the handling station it has to go to the airplane anyway. So there is only a short driving distance to lose.

An IT system for automated dispatching has been introduced at the case company on the basis of the work presented in this paper. The system currently provides decision support for baggage dispatching throughout the airport considered.

Several extensions to the STBRP can be considered. The handling of rush bags is currently considered to be a special, parallel process, i.e. they are not handled by our approach. It would be interesting to consider an integrated approach that includes decision support for rush bags.

Currently, bags and vehicles are not considered for dispatching until they have arrived in the hall. This provides an easy way to run bag scheduling and manual processes in parallel. On the other hand it would be interesting to investigate the effect of using automated scheduling on a larger part of the problem, for instance by considering short term forecasting of bag arrivals, or by also considering more than one vehicle at a time. Integrating forecasting and multiple vehicles would present new and interesing models for the STBRP.

C.7 Acknowledgments

The authors wish to acknowledge the project team at WorkBridge for good discussions and key insights. In particular, the idea of using piecewise linear penalty functions is due to the WorkBridge project team.

References

[1] M. Anker and A. Hämmerle. Applying ant colony optimisation to dynamic pickup and delivery. In R. Moreno-Diaz and F. Pichler, editors, *Computer Aided Systems Theory - EUROCAST 2009*, volume 5717 of *Lecture Notes in Computer Science*, pages 721–728. Springer, 2009.

[2] R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008.

[3] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1998.

[4] G. Berbegliaa, J.-F. Cordeau, and G. Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202:8–15, 2010.

[5] J. Brandao and A. Mercer. The multi-trip vehicle routing problem. *Journal of the Operational Research Society*, 49:799–805, 1998.

[6] J.-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30:105–119, 1997.

[7] D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits. *Transportation Science*, 39:188–205, 2005.

[8] A. Fiat and G. Woeginger. *Online Algorithms*, volume 1442 of *Lecture Notes in Computer Science*. Springer, 1998.

[9] M. Gendreau, F. Guertin, J.-Y. Potvin, and R. Seguin. Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. Technical Report 98-10, 1998.

[10] G. Ghiana and G. Improta. An efficient transformation of the generalized vehicle routing problem. *European Journal of Operations Research*, 122:11–17, 2000.

[11] M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. Subset-row inequalities applied to the vehicle routing problem with time windows. *Operations Research*, 56(2):497–511, 2008.

[12] J. Kozlak, J.-C. Créput, V. Hilaire, and A. Koukam. Multi-agent approach to dynamic pick-up and delivery problem with uncertain knowledge about future transport demands. *Fundamenta Informaticae*, 71:27–36, 2006.

[13] M. E. Lubbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[14] G. Nagy and S. Saldi. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, 162:126–141, 2005.

[15] B. Petersen, D. Pisinger, and S. Spoorendonk. Chvátal-Gomory rank-1 cuts used in a Dantzig-Wolfe decomposition of the vehicle routing problem with time windows. In B. Golden, R. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 397–420. Springer, 2008.

[16] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, 2007.

[17] C. Prins. Efficient heuristics for the heterogeous fleet multitrip vrp with application to a large-scale real case. *Journal of Mathematical Modelling and Algorithms*, 1:135–150, 2002.

[18] J. Renaud, G. Laporte, and F. Boctor. A tabu search heuristic for the multi-depot vehicle routing problem. *Computers and Operations Research*, 23:229–235, 1996.

[19] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Technical Report 04-13, DIKU, University of Copenhagen, 2004.

[20] S. Ropke and D. Pisinger. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171:750–775, 2006.

[21] P. Toth and D. Vigo. An overview of vehicle routing problems. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 1, pages 1–26. SIAM, Philadelphia, 2002.

[22] S. Troschuetz. The troschuetz.random package. .net random number generators and distributions. http://www.codeproject.com/KB/recipes/Random.aspx, 2006.

Paper D

Route Planning for Airport Personnel Transporting Passengers with Reduced Mobility

---

Line Blander Reinhardt, Tommy Clausen and David Pisinger.

Submitted to Transportation Research Part B

Route Planning for Airport Personnel Transporting Passengers with Reduced Mobility

Line Blander Reinhardt, Tommy Clausen, and David Pisinger

Major airports have an average throughput of more than 100,000 passengers per day, some of which will need special assistance. The largest airports have a daily average throughput of more than 500 passengers with reduced mobility. A significant number of people and busses are assigned to provide transportation for the passengers with reduced mobility. It is often necessary for a passenger with reduced mobility to use several different modes of transport during their journey through the airport. Synchronization occurs at the locations where transport modes are changed as to not leave passengers unattended. A description of the problem together with a mathematical model is presented. The objective is to maximize the quality of service by scheduling as many of the passengers as possible, while ensuring a smooth transport with short waiting times. A simulated annealing based heuristic for solving the problem is presented. The algorithm makes use of an abstract representation of a candidate solution which in each step is transformed to an actual schedule by use of a greedy heuristic. Local search is performed on the abstract representation using advanced neighborhoods which modify large parts of the candidate solution. Computational results are reported showing that the algorithm is able to find good solutions within a couple of minutes, making the algorithm applicable for dynamic scheduling. Moreover high-quality solutions can be obtained by running the algorithm for 15 minutes.

D.1 Introduction

At the 31st biggest airport London Gatwick there was a throughput of 32 million passengers in 2009 according to [2]. London Gatwick reports [1] that around 900 passengers with reduced mobility arrived each day.

Such passengers may be passengers returning from vacation with an injury, elderly or weak passengers, blind and deaf passengers, and passengers with other disabilities. In the remaining of this paper we will refer to passengers needing assistance as passengers with reduced mobility (PRM). The support provided for the PRMs may be dedicated transport through the airport, and assistance at boarding. When assisting PRMs through an airport the PRM is picked up at the arriving location, e.g. check-in or gate of arrival, and delivered at the destination location, e.g. arrival hall or gate of departure. It is a service requirement that the PRM is not left alone during the journey from start to end. It may also be possible to assist more than one PRM at a time depending on whether the PRM is in a wheelchair or how well they are able to walk and orient themselves.

In the case studied the objective is to optimize quality of service given the personnel available. Optimizing quality of service is in our case, given a fixed set of personnel and transport objects, to minimize the number of PRMs not delivered and to minimize the total unnecessary travel time used on the journeys. We view the problem of assisting PRMs as a dial-a-ride problem (DARP), which is a generalization of the pickup and delivery problem (PDP). For more details on the dial-a-ride problem (DARP) definition see Cordeau and Laporte [5].

The dial-a-ride Problem (DARP) is NP-hard by reduction from the Hamiltonian cycle problem (Baugh et al. [3]). Normally, the DARP is defined with time windows at either pickup or delivery, but not both, see [5] and [12]. Even though Cordeau and Laporte in [5] argue that having time windows at both ends may be too restricting for the planning, Jaw et al. [12] show that given a pickup time window and a limit on the passenger travel time an implicit time window is imposed on the delivery. Jaw et al. [12] found that explicit definition of the delivery time window improved their algorithm. In the considered problem there can be explicit time windows at both pick up and delivery.

Airports often have several terminals and the transport between the terminals is at the studied airport done in special buses solely for PRMs. Such buses will have a specific location for picking up PRMs at each terminal. Moreover, for aircrafts not located at a gate, the PRM will be transported in a special bus between the gate and the aircraft. Therefore, the pickup and delivery of a PRM is represented as a number of pickup and delivery segments. The airport and airlines require that the PRMs are not left alone at any point during their journey through the airport, and the PRMs are required to be in their assigned flight seat at a fixed pre-specified time before departure. However, the PRM may be left alone for a while before boarding at the departing terminal in a supervised area.

Between each pick up and delivery of a PRM the transport object delivering

the PRM must meet the transport object picking up the PRM. This vehicle synchronization is called a temporal dependency, therefore the problem is a dial-a-ride problem with temporal dependencies (DARPTD). The concept of synchronization in routing was used by Ioachim et al. [11] for the fleet assignment problem and later expanded to the more general temporal dependencies by Dohn el al. [9]. In pickup and delivery problems the similar problem of cross docking has been considered, which has a transfer of goods between vehicles at the synchronized points. The pickup and delivery with cross docking is used in supply chain and planning city logistics systems [4], [7]. Pickup and delivery with cross docking was studied by Wen et al. [15]. In the cross docking problems the cross docking is optional for the vehicles. This is not the case in the problem of assisting PRMs at an airport, as the cross-docking points for each PRM are known and fixed. The synchronization constraints and the objective separates the routing of transport objects transporting PRMs in airports from the rich pickup and delivery problem described in [13].

In this paper we have constructed a local search heuristic for the specific problem based on simulated annealing. The algorithm makes use of an abstract representation, which is transformed to an actual schedule by use of a greedy heuristic. Local search is performed on the abstract representation using large neighborhoods. In each iteration the resulting candidate solution is evaluated, and accepted according to the standard criteria in simulated annealing. Computational results are reported showing that the algorithm is able to construct high-quality solutions in 15 minutes.

This paper is organized as follows. Section D.2 contains a detailed problem description to ensure a thorough understanding of the operational process. In Section D.3 a mathematical model of the problem is presented. Section D.4 presents the solution methods used. Section D.5 contains the specifications of the data instances received. In Section D.6 the tuning of the parameters for simulated annealing us described. Section D.7 contains the results of the solution method applied to the real-life instances received. In Section D.8 there is a discussion on the results and future work.

D.2 Problem Description

We will term the considered problem the *Airport passenger with reduced mobility transport problem* (APRMTP). The APRMTP has been defined in cooperation with a service company providing the assistance for PRMs at a major transit airport. The company does not deliver service in the entire airport but in the majority of the airport. The company has 120 employees assisting between 300

and 500 PRMs through the airport each day. The employees have a prespecified working area such as a specific terminal, driving between terminals bus stop locations or driving between aircrafts and gates. A worker assigned to one area may not move into another area. Therefore, the journey of the PRM is split up into a pickup and delivery for each of these areas. We call the pick up and delivery in a specific area for a segment and the ordered set of segments of a given PRM for a journey. On average there are three segments per PRM, given the 300 to 500 PRMs each shift we get a total of between 900 and 1500 pick up and delivery segments. This also includes assistance when boarding, which we have represented as a pickup and delivery request with special conditions. We will in the remainder of this paper refer to the boarding assistance as embarkment. The employee assigned to an embarkment cannot go to another location between pickup and delivery of the embarkment segment. However, an employee may assist as many PRMs embarking on to the same flight as their capacity allows.

It should be noted that all of the pickup and delivery locations for every segment of the journey are predetermined.

As mentioned in the previous section, the PRM may not be left alone except at special supervised areas located in the departing terminal. This means that the employee delivering the PRM to a bus must wait with the PRM for the bus to arrive at the bus stop before being able to initiate the next task. The bus also has to wait for the employee to come and pick up the delivered PRM before continuing the route.

The company wants to make sure that they deliver the best service possible with the given number of employees and the current employees working area assignments. The PRMs are split into two categories: Those who are prebooked and those who are immediate. The prebooked PRMs ordered the service when purchasing the ticket or at least days in advance. Immediate PRMs requests the service at check-in, and therefore may only be known half an hour in advance for PRMs arriving on flights or at check-in for passengers checking in at the airport. It is not always possible for the company to assist all PRMs and in such cases the prebooked PRMs have higher priority. The company also wishes to minimize unnecessary time the PRM spends on the journey segments. Unnecessary time could be time spent waiting to be picked up by the employee working in the area of the succeeding segment or extra travel time caused by picking up or delivering other PRMs before being delivered. Note, that the time spent at the supervised area of the departing terminal is not included in the service evaluation. The unnecessary time spent on the segments we call excess time. The problem is then to route the employees on foot or in their assigned vehicle so that the service quality is maximized. Clearly, when minimizing the overall traveling time there is a risk of having a few PRMs with very large traveling times. Therefore, it is important to limit the traveling times of the different segments so that the

journey never becomes very unsatisfactory.

Many additional constraints concerning the pick-up and delivery times, assistance at embarkment and transport to and from aircrafts not located at a gate, are imposed by the airport and airlines. Such constraints are

- an arriving PRM must be picked up exactly upon arrival
- a terminal transfer takes place on bus between the bus stop locations of the terminals
- the PRM may not be left unsupervised
- the PRM must not use more than 30 minutes of excess time on each segment
- embarkment takes 20 minutes and can not start earlier than 60 minutes before departure.
- the PRM must be seated in the plane exactly 20 minutes before departure.

The last two items mean that embarkment can not start later than 40 minutes before departure and even earlier when plane is parked away from gate.

Note, that the person assisting the PRM through embarkment will not be able to leave the PRM until 20 minutes before departure if the aircraft is located at gate or before the PRM is picked up by special vehicle if the aircraft is located away from gate. In the latter case the special vehicle cannot leave the PRM before 20 minutes before departure.

The different transportation forms such as vehicles and assistance on foot have different capacities. Moreover the PRMs are assigned a volume depending on their disability. For example it is very hard for one person to push two wheelchairs however, two hearing or sight impaired persons can be assisted by the same employee.

D.2.1 Example of a journey of an PRM

The most complex example of a journey is the case where a PRM makes a transit from an arriving aircraft not located at the gate to an departing aircraft in another terminal not located at the gate. In such a case the segments of the journey are as follows (see also Figure D.1)

1. The PRM is picked up by a vehicle at the arriving aircraft on the exact time of arrival (it is a requirement that the PRMs are picked up exactly upon arrival) and delivered at gate.
2. The PRM is picked up at the gate by an employee. The bus delivering the PRM in segment 1 cannot leave before the PRM is picked up. The PRM is delivered at the bus stop for the special inter terminal busses.
3. The PRM is picked up by a special inter terminal bus at the bus stop and delivered to the bus stop at the terminal of the departing flight. Again the employee of segment 2 cannot leave before the bus arrives for pickup.
4. The PRM is picked up by an employee and delivered to the gate of departing flight. Again the bus in segment 3 cannot leave before the employee arrives.
5. The PRM is assisted by an employee through embarkment at gate. This task takes 20 minutes. For the time between segments 4 and 5 the PRM can be left at a special supervised lounge.
6. The PRM is picked up at the gate by a vehicle and delivered to the plane exactly 20 minutes before the departure time of the flight. Again the employee of segment 5 cannot leave before the vehicle arrives for pickup.

We say that such a journey has six segments. All transit journeys are formulated as an ordered subset of these segments. Non-transit PRMs are either picked up or delivered to a public area in the same terminal as respectively the arriving or departing flight.

The algorithm is used in a dynamic setting, where immediate PRMs arrive continuously and disruptions in the daily plan such as flight delays frequently occurs. Therefore, the company desires to receive a solution to the problem within a couple of minutes. We do not consider robustness and break times. However, robustness can be obtained by introducing buffer time in the time to get from one location to another and by altering the set of employees available and breaks maybe included by splitting the shift of an employee into several shifts.

D.3 Mathematical formulation

When describing the model it is important to bear in mind that the journey of each PRM is a set of pick up and deliveries called segments. This means

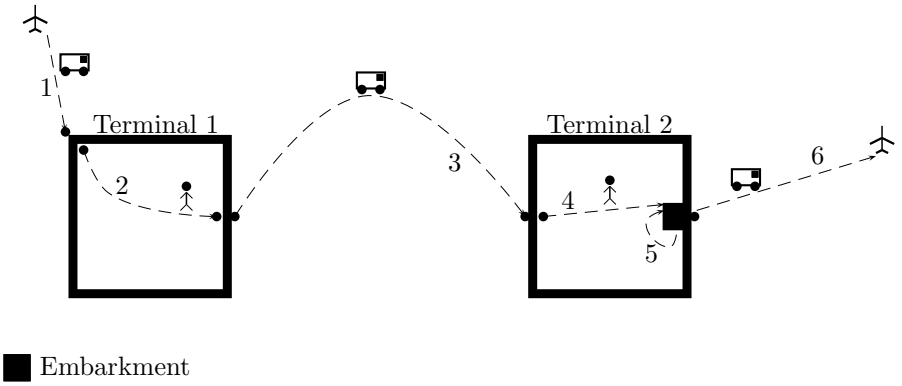


Figure D.1: Illustration of a journey with six segments: (1) The PRM is picked up by a vehicle at the arriving aircraft and brought to the gate, (2) The PRM is brought to the inter terminal bus. (3) The PRM is transported by a inter terminal bus from Terminal 1 to Terminal 2, (4) The PRM is brought to the gate of departing flight, (5) The PRM is assisted through embarkment at gate, (6) The PRM is delivered to the aircraft.

that a PRM is picked up and delivered if all the segments of the journey are handled. Therefore, we must for each PRM make sure that all their segments are assigned before we consider them delivered. As common in dial-a-ride problems with a heterogeneous fleet each segment is represented by a pickup vertex and a delivery vertex specific to that segment. There is a location inside each area where the employees start and end their shift.

D.3.1 Graph representation

There is as mentioned earlier a vertex for each origin and destination of a journey segment. The location of all the vertices in a journey are pre determined. Since we already know which transport group is assigned to each vertex we can generate a disjoint graph for each transport group. Each graph has its own depot where the transport objects starts and end their work. These graphs are only "virtually" connected by the connection between segments of a journey. For each terminal we have a directed graph connecting the vertices that must be serviced by foot personnel working in the given terminal. The busses transporting PRMs between terminals have a directed graph of their pick up and delivery vertices. The vehicles transporting PRMs from gates to airplanes have

a directed graph connecting their pickup and delivery vertices. Connections between pick up and delivery vertices, which are infeasible due to their time windows, are removed from the graph.

D.3.2 Mathematical model

Given the following sets:

K	The set of transport objects. Contains all vehicles and persons on foot
R	The set of segments. Contains all the segments of all the journeys
R_p	The set of segments. Contains all the segments for PRM p
B	The set of prebooked PRMs
C	The set of all PRMs

F	The set of departing flights
V	The set of pick up and delivery points/vertices
$V*$	The set of pickup and delivery vertices and depots/bases
V_f	The set of embarkment vertices for flight $f \in F$
P	The set of pickup vertices
D	The set of delivery vertices
E	The set of edges connecting the elements in $V*$
λ_p	The set of vertex pairs (i, j) where i is the delivery vertex of the segment right before the segment with pickup vertex at j on a journey for PRM p
δ	The set of vertex pairs that must be synchronized for handover

Each segment has a start o_d and a destination t_d and the set V is all of the different o_d and t_d vertices. Each work area has a starting point v_0 and an end point v_e representing the location, where the transport objects start and end the day, by two vertices.

We define the following parameters:

M_b	The penalty for not transporting a prebooked PRM
M_n	The penalty for not transporting an immediate PRM
$t_{o_d t_d}$	The minimum time needed to deliver segment $d \in R$
t_{ij}^k	The minimum time it takes to go from i to j on transport k
l'_j	The change in load at vertex $j \in V$
H	The maximum excess time allowed to be used on a segment. Here $H = 30$
C_k	The capacity of transport object $k \in K$
M	A big constant being at least as large as the shift length
M_s	A big constant larger than the largest number of segments in a PRM
M_l	A big constant at least as large as the biggest capacity plus the largest volume possible for a PRM
a_i	The release time at vertex $i \in V*$
b_i	The due time at vertex $i \in V*$

We use the following variables:

s_i^k	the time when transport object k leaves vertex i
ϕ_p	An indicator variable indicating if a PRM $p \in C$ has a segment not assigned. ϕ_p is 0 if all segments of p are assigned and 1 otherwise
x_{ij}^k	An indicator variable indicating if the edge (i, j) is used by object k. x_{ijk} is 1 if the edge is used by k and 0 otherwise
l_i^k	the load on transport object k when leaving vertex i

As objective we have chosen an linear weighted combination of assigning as many PRMs as possible and minimizing the total excess time the PRMs spend on their journey:

$$\min \quad \sum_{d \in R} \left(\left(\sum_{k \in K} s_{t_d k} - s_{o_d}^k \right) - t_{o_d t_d} \right) + \sum_{p \in B} (1 - \phi_p) * M_b + \sum_{p \in C \setminus B} (1 - \phi_p) * M_n \quad \text{(D.1)}$$

s.t.

$$\text{(P and D)} \quad \sum_{i \in V} x^k_{io_d} - \sum_{i \in V} x^k_{it_d} = 0 \qquad j \in R,\, k \in K \qquad \text{(D.2)}$$

$$\text{(balance)} \quad \sum_{j \in V} x^k_{ij} - \sum_{j \in V} x^k_{ji} = 0 \qquad i \in V,\, k \in K \qquad \text{(D.3)}$$

$$\text{(start)} \quad \sum_{j \in V} x^k_{v_0 j} = 1 \qquad k \in K \qquad \text{(D.4)}$$

$$\text{(end point)} \quad \sum_{j \in V} x^k_{j v_e} = 1 \qquad k \in K \qquad \text{(D.5)}$$

$$\text{(P} \rightarrow \text{D)} \quad s^k_{t_d} - s^k_{o_d} \geq 0 \qquad k \in K,\, d \in R \qquad \text{(D.6)}$$

$$\text{(Complete)} \quad M_s \phi_p + \sum_{d \in R_p} (1 - x^k_{o_d j}) \geq 0 \qquad k \in K\, p \in C \qquad \text{(D.7)}$$

$$\text{(Timelimit)} \quad s^k_{t_d} - s^k_{o_d} - t^k_{o_d t_d} \leq H \qquad k \in K,\, d \in R \qquad \text{(D.8)}$$

$$\text{(Connect)} \quad s^k_i + t^k_{ij} + M(x^k_{ij} - 1) \leq s^k_j \qquad k \in K,\, (ij) \in E \qquad \text{(D.9)}$$

$$\text{(Handover)} \quad \sum_{k \in K} s^k_i = \sum_{k \in K} s^k_j \qquad (i,j) \in \delta \qquad \text{(D.10)}$$

$$\text{(Journey)} \quad \sum_{k \in K} s^k_i \leq \sum_{k \in K} s^k_j \qquad (ij) \in \lambda_p \qquad \text{(D.11)}$$

$$\text{(Release)} \quad a_i \leq s^k_i + a_i(1 - \sum_{j \in V*} x^k_{ij}) \qquad i \in V*,\, k \in K \qquad \text{(D.12)}$$

$$\text{(Due)} \quad b_i \geq s^k_i + b_i(1 - \sum_{j \in V*} x^k_{ji}) \qquad i \in V*,\, k \in K \qquad \text{(D.13)}$$

$$\text{(Load)} \quad l^k_i + l'_j - M_l(x^k_{ij} - 1) \leq l^k_j \qquad (ij) \in E,\, k \in K \qquad \text{(D.14)}$$

$$\text{(Capacity)} \quad l^k_i \leq C_k \qquad i \in V,\, k \in K \qquad \text{(D.15)}$$

$$\text{(Emb)} \quad \sum_{k \in K} x^k_{ij} = 0 \qquad j \in V \setminus V_f, i \in P \cap V_f, f \in F \qquad \text{(D.16)}$$

$$\text{(Emb load)} \quad \sum_{j \in V \setminus V_f} (C_k x^k_{ij} - l^k_j) \geq 0 \qquad k \in K, i \in D \cup V_f, f \in F \qquad \text{(D.17)}$$

$$\text{(Variables)} \quad x^k_{ij} \in \{0,1\} \qquad k \in K,\, (ij) \in E \qquad \text{(D.18)}$$

$$\phi_p \in \{0,1\} \qquad p \in C \qquad \text{(D.19)}$$

$$s^k_i \in \mathbb{R}^+_0 \qquad i \in V \cup \{p_k\},\, k \in K \qquad \text{(D.20)}$$

$$l^k_i \in \mathbb{Z}^+_0 \qquad i \in V,\, k \in K \qquad \text{(D.21)}$$

The objective function (D.1) sums all the excess time used on the segments and adds a penalty if a PRM is not delivered. The penalty depends on whether PRM p is prebooked ($p \in B$) or immediate ($p \in C \setminus B$). Constraints (D.2) ensure that for each segment any PRM picked up is also delivered. Constraints (D.3) ensure that transport objects leaves all pickup or delivery vertices they enter. Constraints (D.4) ensure that all transport objects leaves their base. Constraints (D.5) ensure that all transport objects return to their base. Constraints (D.6) ensure that on each segment a PRM is picked up before it is delivered.

Constraints (D.7) ensure that any PRM with at least one segment not assigned generates exactly one penalty in the objective. Constraints (D.8) ensure that the excess time used on segment d does not exceed H. Constraints (D.9) ensure that if an edge (i, j) is used the time vertex j is visited is greater than the time vertex i is visited plus the travel time on edge (i, j). Constraints (D.10) ensure that delivering transport object meets pickup transport object for at delivery pickup handover vertex pair in δ. Constraints (D.11) ensure that the segments of the journey are completed in the right order. Constraints (D.12) and (D.13) ensure that the segments are started and ended within their given time window. Constraints (D.14) ensure that the load is updated when a PRM is picked up or delivered. Note that since load is increased for any a pickup vertex in V then constraints (D.14) together with constraints (D.9) ensure a connected route. This is true under the general assumption that the transport from pickup to the delivery point is always greater than zero. Constraints (D.15) ensure that the capacity is not exceeded. Constraints (D.16) and (D.17) enforce the embarkment conditions of only starting embarkment tasks on the same flight before completing a embarkment segment. Constraints (D.16) only allow edges going from a pickup vertex of an embarkment segment to vertices of embarkment on the same flight. Constraints (D.17) ensure that when using and edge between an embarkment vertex and any vertex not belonging to an embarkment request for the same flight the load must be zero.

D.4 Solution method

The solution method we present is a greedy insertion heuristic combined with simulated annealing.

In the survey by Cordeau and Laporte [5] from 2007 a list of some of the methods used for the dial-a-ride problem with multiple vehicles is provided. In this list the only exact methods are a branch and cut method optimizing on vehicle travel cost by Cordeau [6] and an improvement on this method by Ropke et al. [14]. The exact method has been tested on a maximum 96 requests and 8 vehicles, which was solved in 71 minutes.

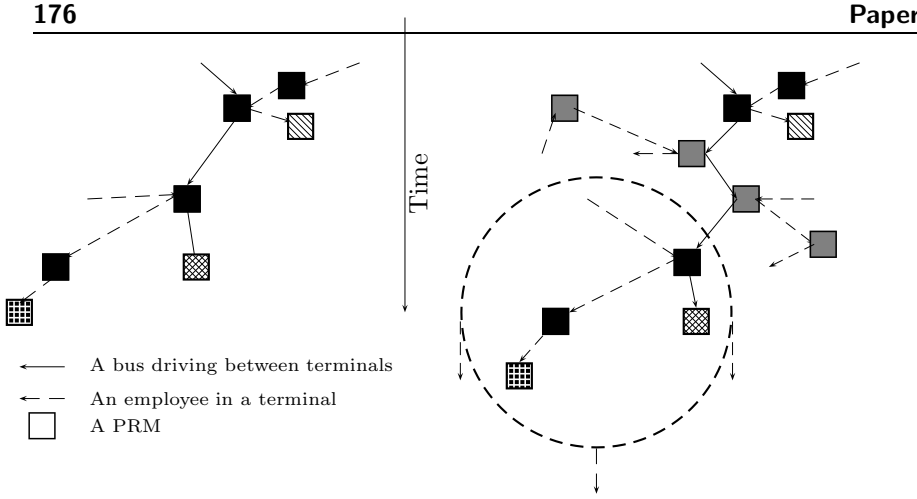
The dial-a-ride problems are usually solved by heuristics as the problems are often real-life problems. Real life problems generally contains some additional constraints, which can be complicating and the objective varies. Moreover in real-life there can be constraints or desires not defined in the problem, which arises after the problem definition. Due to this an optimal solution might actually not be the best solution for the users.

Since the problem covered here is a dial-a-ride problem with complicating synchronization and embarkment constraints it is natural to consider heuristic solution methods. Moreover since the requirement is to solve instances with between 900 and 1500 requests within 2 minutes a heuristic method seems to be the only option.

Jaw et al. [12] in 1986 reports finding a good solution to their dial-a-ride problem on an instance with 2617 requests and 28 vehicles using an insertion sort method. Given the machines available in 1986 the solution is found quickly and the method would on the machines available today satisfy the solution time requirement. Other heuristic methods that are able to solve dial-a-ride problems with a large number of requests are Diana and Dessouky [8] using regret insertion solving problems with a 1000 requests and Xiang et al. [16] using a local search heuristic solving problems with 2000 requests.

The synchronization constraints present in the airport PRM transport problem (APRMTP) do add some complexity to the generation of feasible solutions and the calculation of the objective value. When a segment is inserted in a route it may have influence on not only the travel times of the later segments in the route, but also the other segments of the PRM and the segments of their routes and so forth. This means that every time a segment end and start time is changed it may generate a cascade of changes on related segments. Therefore, when checking for feasibility one may, in worst case, have to evaluate the feasibility of all the segments in the problem . The same is true when calculating the objective as the insertion of a segment may influence the travel time on all remaining segments in all the routes. However, together with a constraint on the maximum excess time allowed on each segment it may also constrain the problem significantly. An example of this is that the synchronization constraint reduces the number of possible feasible solutions.

We have constructed an insertion heuristic for the initial solution, which is described in the next section and later used in a simulated annealing scheme to improve solutions. The greedy insertion heuristic will given an ordered list lead to a deterministic solution found with a search for best insertion spot while the simulated annealing broadens the search by randomly selecting a neighborhood from a large neighborhood space and accepting solutions with a worse objective value. This method is similar to GRASP [10] as there at each iteration is a greedy construction. However, to avoid the in this case hard task of evaluating candidates we have a fixed order. The routes in the constructed solution are in the APRMTP very interdependent and therefore it would be very difficult for a local search to find better solutions using neighborhoods on the constructed solution. Instead of performing a local search on the constructed solution as done in GRASP we perform the local search on the fixed candidate lists given for the previous solution. Because of this reverse execution of the GRASP



(a) Before insertion (b) After insertion of gray PRM

Figure D.2: A section of the routes when inserting the gray PRM. The part of the graph inside the dashed circle is moved to a later time because of propagated delays caused by the insertion of the gray PRM. The patterned PRMs are the next PRMs to be handled by the employee or bus.

structure the algorithm presented could be described as a reversed GRASP.

Figure D.2 shows how PRM segments are moved when inserting a PRM segment into the route represented by the solid line.

D.4.1 Insertion heuristic

To quickly find a feasible solution we have created a greedy insertion heuristic (GIH). The heuristic takes two lists of PRMs one containing the prebooked PRMs and one containing the immediate PRMs. The insertion heuristic inserts first the PRMs in the prebooked list then the PRMs in the immediate list by going through the list in sequential order. The reason for this is that it is very important for the service provider to serve the prebooked PRMs.

We have sorted the lists by earliest pickup time of the PRM, starting at the earliest. For each PRM the segments are inserted in the order they appear in the journey and the next PRM is not inserted before all segments of the previous PRM are inserted.

For each segment only the set of transport objects working in the graph containing the given segment are investigated for an insertion place. The segment

is inserted in the place and transport object where it creates the least increase in the total excess time. We only allow an insertion to push the time of the other segments forward. Therefore, when checking for feasibility we only need to go through the segments with larger delivery times.

Usually when minimizing the route cost as in pickup and delivery problems it is possible to calculate the new objective by the difference in the time introduced by the insertion and removal however, in our case we did not do this as we found it too complicating when minimizing excess time and number of undelivered PRMs especially when there are propagating delays induced by the synchronization constraints.

GIH(P_1, P_2)

```
1:  for each p PRM first in list P1 then the list P2  do
2:    for each segment s in p do
3:      for each employee e serving s  do
4:        for each vertex v1,v2 in e in the possible time interval for s do
5:          if load and time feasible insert s_start before v1 and s_end before v2
            then
6:            calculate total excess time for all inserted segments;
7:          end if
8:        end for
9:        Select s1 and s2 where the least excess time is generated;
10:     end for
11:     if insertion was not possible then
12:       Delete already inserted segments of p;
13:       Register p as not inserted;
14:     else
15:       insert s in the e where the least excess time is generated;
16:     end if
17:   end for
18: end for
```

In the pseudo code of **GIH** the lines 1 and 2 contributes to the complexity of **GIH** with the total number of requests $|R|$, which we here call n. The combinations that occur in line 3 and 4 can be n^2. Checking for feasibility in line 5 is done by a depth first search which at most goes through $2n$ vertices updating their times and the edge load. The calculation of total excess time of all inserted segments in line 6 is done by adding up excess time of all inserted segments, which is at most n. Therefore, the asymptotic running time of the greedy insertion sort is $O(n^4)$.

However, in the test cases provided by the company the time windows are quite tight and not all employees are available in the area of a given segment therefore there are often only a few locations where feasibility is actually checked and excess time calculated.

D.4.2 Simulated annealing

A Simulated annealing algorithm using the two lists of PRMs as an abstract representation was implemented. The initial solution is the greedy insertion heuristic on the two lists of PRMs, sorted by earliest possible start time. At each iteration in the simulated annealing algorithm a number of moves takes place to obtain a candidate solution x from the current solution x'.

The moves are made in the two PRM lists, which are then converted to a solution by the insertion heuristic. The moves are as follows:

- moving a not assigned PRM a random number of places forward in their respective lists
- to swap the place of two PRMs randomly selected within the same list.

Note, that the prebooked PRMs and immediate PRMs will always remain in their respective lists. The lists are when the moves are completed converted by the greedy insertion heuristic into a schedule.

Let the objective function be defined as $f(x)$ for a solution x. If $f(x) \leq f(x')$ then x is accepted. Otherwise we accept the solution with probability:

$$\exp^{f(x')-f(x)/T} \tag{D.22}$$

Details on the selected values for the temperature T will be covered in Section D.6 on tunings, The temperature is decreased by a selected factor at each iteration. This decreasing factor is also called the cooling rate. The large neighborhood described and the acceptance probability allow the algorithm to escape local minima.

D.5 Data Instance and other parameter values

We were from the service company given a list of almost 5000 PRM request with information about the type and the position of the origin and destination. A

travel time between the locations for the different transport forms was generated from this information. We have received 12 test cases from the company covering dates September 20 to October 1, 2009. These data sets contain between 374 and 555 PRMs each day. Some of the PRMs in the data set were removed before running the tests due to corrupted data for the PRM or that too little time was available for the PRMs journey so that a solution transporting the PRM can not exist. This resulted in sets of between 353 and 495 PRMs. The data sets each had a set of employees for the given day and their assigned terminal or vehicle. The number of employees assigned on a day was around 120. The employees were assigned to 6 different terminals and 2 different bus types. For each bus type there is a type specific area of operation.

The capacities of the employees has been settled with the ground handling company as:

- Employee assisting inside terminal has capacity 4
- Bus between terminals has capacity 24
- Bus between gate and aircraft has capacity 18

For each PRM there is given a start time, an end time, a start location, an end location and a PRM type. There where six different types of PRMs in the data sets for each of the types we have assigned a volume as follows:

WCHC Cannot walk or stand. Needs wheel chair. Volume 3

WCHS Cannot walk up or down stairs. Volume 2

WCHR Cannot walk long distances. Volume 2

BLND Passenger is blind. Volume 2

DEAF Passenger is deaf. Volume 2

ASS Passenger cannot orient them selves. Volume 2

From the PRM data and the rules given by the company for the journey we generated the segments for each PRM given the arrival and departure location of the PRM. For each shift between 900 and 1500 segments where generated.

D.6 Tuning

Since our insertion algorithm has a complexity of $O(n^4)$, and the time allowed to solve the overall problem is limited to a few minutes, the number of iterations, which can be investigated in simulated annealing given the size of the problem instances is limited to a few hundreds. Hence, frequently the problem cases contain more PRMs than iterations performed in simulated annealing. Therefore, we consider the possibility for making several moves at each iteration. The moves are relocations in the list and thus doing more moves does not influence the running time significantly. However, the neighborhood becomes much larger and the previous solutions may be ruined by a large number of moves.

We first test the combination of different number of moves with different cooling rates given an fixed initial temperature. From the tuning tests a good combination of cooling rate and number of moves is selected and used in the test of the different possibilities for the initial temperature. The tuning is done of a solution time of 2 minutes as this is the requirement for the solution time given by the users.

The initial temperature is adjusted so that the probability of selecting a solution, which is exactly t percent greater than the initial solution is 50%. For finding the best combination of the cooling rate and the number of moves, we have fixed the initial temperature so that a solution 5% worse than the initial solution must initially be accepted with 50% probability. This means that given an initial solution x and the temperature parameter of t then the initial temperature T is calculated as follows:

$$T = -tx/\log(0.5) \tag{D.23}$$

We made the choice of including the initial objective value in the generation of the initial temperature so that the variance in the size of the different problems does not influence the acceptance rate.

The three test cases used for tuning were randomly selected from the data sets received. Note that the maximum excess time on a segment $H = 30$ for all tests runs as this generally matches the requirement of the airports.

Case	prms	deleted	time	init NAP	init NAI	init sol
20090920	353	21	120	0	4	3566
20091001	474	45	120	1	1	4401
20090926	374	27	120	0	0	1782

Table D.1: The test cases used for tuning, with the results from the greedy insertion heuristic.

In Table D.1 we report the characteristics of each data instance and the values

of the initial solution constructed with the greedy insertion heuristic on the lists, where the PRMs are sorted by earliest arrival time. The name of the test case is given in column one. In column two the number of PRMs in the test case is provided and in column three the number of PRMs that was deleted from the initial data set due to corrupted data. This means that column two and column three gives the number of PRMs in the initial data set. In column four the time in seconds allowed for simulated annealing is given. In all of the tuning cases we have used a solution time limit of 2 minutes. Column five reports the number of unassigned prebooked PRMs (NAP) in the initial solution and column six reports the number of unassigned immediate PRMs (NAI). In column seven the value of the initial solution is reported.

testcase	coolrate	moves	average sol	av ite	stdD	av NAP	av NAI	best sol	gap
	0.5	4	2655.2	326.5	179	0	2.3	2491	25.5%
	0.8	4	2732.2	325.9	187	0	2.5	2457	23.4%
	0.9	4	2720.2	326.7	242	0	2.4	2426	23.7%
	0.95	4	2753.5	324.5	201	0	2.5	2511	22.8%
	0.99	4	2912.2	322.1	239	0	2.5	2605	18.3%
	0.5	12	2495.8	315.8	57	0	2.0	2396	30.0%
	0.8	12	2496.3	316.7	58	0	2.0	2396	30.0%
20090920	0.9	12	2510.4	315.5	34	0	2.0	2461	29.6%
	0.95	12	2500.0	310.1	55	0	2.0	2431	29.9%
	0.99	12	2609.3	306.3	52	0	2.0	2541	26.8%
	0.5	20	2521.5	306.1	79	0	2.0	2351	29.3%
	0.8	20	2505.6	309.9	47	0	2.0	2418	29.7%
	0.9	20	2515.2	308.8	33	0	2.0	2450	29.5%
	0.95	20	2495.0	301.2	53	0	2.0	2437	30.0%
	0.99	20	2564.3	298.7	46	0	2.0	2464	28.1%
	0.5	4	2691.7	125.3	198	0	0.4	2461	38.8%
	0.8	4	2796.7	125.4	130	0	0.8	2521	36.5%
	0.9	4	2794.8	123.2	120	0	0.5	2543	36.5%
	0.95	4	2737.5	120.0	187	0	0.2	2319	37.8%
	0.99	4	3149.8	117.7	205	0	0.7	2869	28.4%
	0.5	12	2838.9	120.8	111	0	0.3	2736	35.5%
	0.8	12	2742.0	115.1	161	0	0.1	2510	37.7%
20091001	0.9	12	2723.3	114.4	190	0	0.3	2472	38.1%
	0.95	12	2749.2	107.7	126	0	0.3	2598	37.5%
	0.99	12	2893.6	104.1	130	0	0.2	2677	34.3%
	0.5	20	2891.6	113.2	131	0	0.1	2789	34.3%
	0.8	20	2834.0	110.4	141	0	0.2	2635	35.6%
	0.9	20	2820.9	106.1	184	0	0.3	2459	35.9%
	0.95	20	2831.5	103.3	136	0	0.4	2644	35.7%
	0.99	20	3048.1	100.0	129	0	0.6	2842	28.1%
	0.5	4	1442.6	255.1	40	0	0	1378	19.0%
	0.8	4	1427.9	256.5	50	0	0	1368	19.9%
	0.9	4	1428.5	254.8	41	0	0	1352	19.8%
	0.95	4	1480.8	254.8	48	0	0	1395	16.9%
	0.99	4	1563.0	250.1	80	0	0	1437	12.3%
	0.5	12	1482.0	248.0	50	0	0	1425	16.8%
	0.8	12	1484.3	245.1	42	0	0	1420	16.7%
20090926	0.9	12	1431.5	245.2	52	0	0	1322	19.7%
	0.95	12	1491.7	241.6	70	0	0	1380	16.3%
	0.99	12	1508.8	236.1	78	0	0	1399	15.3%
	0.5	20	1508.4	243.0	50	0	0	1400	15.4%
	0.8	20	1496.7	240.2	57	0	0	1424	16.0%
	0.9	20	1490.5	237.1	73	0	0	1370	16.4%
	0.95	20	1471.1	234.5	55	0	0	1347	17.4%
	0.99	20	1520.9	228.9	55	0	0	1422	14.7%

Table D.2: Tuning cooling rate and number of moves

In Table D.2 we have fixed the temperature parameter $t = 0.05$. The tuning is done for each of the test instances described in Table D.1. For each test case and each combination of cooling rate and number of moves given in respectively column two and three the algorithm is run ten times and the average solution of the ten runs is reported in column four. The average number of iterations completed within the two minutes each run lasted is reported in column five.

The standard deviation of the solutions is reported in column six. The average number of unassigned prebooked PRMs and immediate PRMs is reported in respectively column seven and eight. In column nine the best solution of the ten runs is given. The gap reported in column ten is the percent wise gap between the initial solution and the average solution found by simulated annealing calculated as:

$$gap = \frac{initsolution - averagesolution}{initsolution} \cdot 100$$

Note that by using the average solution reported in column three the gap represents the expected improvement for a single 2 minute run of the simulated annealing algorithm.

Analyzing the results in Table D.2 using ranking of the gap for each of the three data sets and comparing their ranks we have chosen the value 0.9 for the cooling rate and 12 moves at each iteration. The selected values for cooling rate and number of moves are used in the tuning test on values for the initial temperature shown in Table D.3.

Case	temperature	av best sol	av it	stdD	av NAP	av NAI	best sol	gap
20090920	1%	2490.3	317.0	53	0	2.0	2379	30.2%
	5%	2510.4	315.5	34	0	2.0	2461	29.6%
	10%	2585.3	313.4	146	0	2.2	2454	27.5%
	15%	2542.9	312.6	36	0	2.0	2462	28.7%
20091001	1%	2812.2	107.0	175	0	0.4	2470	36.1%
	5%	2723.3	114.4	190	0	0.3	2472	38.1%
	10%	2717.9	109.2	130	0	0.2	2538	38.2%
	15%	2774.5	108.8	234	0	0.3	2428	37.0%
20090926	1%	1470.3	248.3	64	0	0	1387	17.5%
	5%	1431.5	245.2	52	0	0	1322	19.7%
	10%	1473.8	241.2	54	0	0	1370	17.3%
	15%	1486.4	242.0	44	0	0	1384	16.6%

Table D.3: Tuning initial temperature

The columns in Table D.3 are structured the same way as Table D.2, except that the columns reporting cooling rate and moves in Table D.2 are replaced by a single temperature column in Table D.3. The results in Table D.3 show that the best initial solution is obtained bychoosing the point where a solution will be accepted with 50% probability as 5% under the initial value. These values will be used for the tests in Section D.7. The values found in this section for the simulated annealing determines the intensification and diversification of the heuristic. The high number of moves at each iteration and the acceptance probability ensures diversification while the cooling rate generates an intensification. The start temperature determines the start diversification generated by the acceptance probability.

D.7 Test Results

We have received 12 test cases from the service company covering dates September 20 to October 1, 2009. Unfortunately, the company does not have records on how the PRMs actually where schedule for those test cases, as the system mainly takes care of registering the PRMs arriving, what segments needs to be completed and which employees are available for completing them. Hence, we cannot directly compare our schedules with the historic data.

It should be noted that in the data sets a little more than half of the PRMs are prebooked. Three of the twelve data sets has been used for tuning, however, we still include them in this test section. The test are run on a computer with a 64 bit Intel Xeon 2.67 GHz CPU.

All the simulated annealing tests reported in this section has been run with:

- Temperature: we use the factor $t = 0.05$ to adjust the temperature T according to (D.23).
- Cooling rate: 0.9
- Moves at each iteration: 12

Note that temperature is calculated given an initial solution x and the temperature parameter of t as described in equation (D.23).

In Table D.4 we report the results of running the greedy insertion heuristic on the data set where the PRMs are sorted by earliest arrival time for all 12 data sets. For each data set the excess time allowed on a segment is again $H = 30$ minutes corresponding to airport requirements.

In column one the name of the data set is described by the date of the shift. We report how many PRMs in the given data set we had to remove due to corrupted data for the PRM or that too little time was given for the PRMs journey so that a solution transporting the PRM can not exists. Note that the number of PRMs reported in column two is the number of PRMS after the removal of the deleted PRMs reported in column three from the initial set. The number of prebooked passengers not assigned (NAP) and not assigned immediate passengers (NAI) in the initial solution is reported in respectively column four and five. Finally the initial solution retrieved from the greedy insertion heuristic on the PRM lists sorted by earliest arrival time is reported in column six.

Case	prms	prms deleted	NAP	NAI	sol
20090920	353	21	0	4	3566
20090921	391	35	7	5	11430
20090922	428	23	0	5	5162
20090923	361	42	0	0	2001
20090924	432	33	0	3	4062
20090925	438	46	0	2	3500
20090926	374	27	0	0	1782
20090927	397	32	0	2	3095
20090928	378	23	0	0	2208
20090929	419	37	0	1	3259
20090930	495	60	7	9	13443
20091001	474	45	1	1	4401

Table D.4: The results of the insertion heuristic run on the different data sets. Note, that this is also the initial solution used by the simulated annealing heuristic.

From Table D.4 it can be seen that for 9 of the 12 instances there is no prebooked PRM rejected. The number of rejected immediate PRMs is also quite low compared to the total number of PRMs when solving the problem using just the greedy insertion heuristic.

The results in Table D.4 are used for evaluating the test results of the simulated annealing algorithm presented in Table D.5.

Case	# runs	time (s)	av sol	av it	stdD	av NAP	av NAI	best sol	gap
20090920	10	120	2504.6	279.5	61	0	2.0	2423	29.8%
20090920	10	300	2426.9	767.6	52	0	2.0	2361	31.9%
20090920	3	3600	2223.3	7812.3	20	0	2.0	2197	37.7%
20090921	10	120	9026.0	209.0	369	5.6	3.6	8227	21.0%
20090921	10	300	8658.3	499.4	430	5.6	3.0	8112	24.2%
20090921	3	3600	7713.7	4063.3	663	5.3	2.3	7205	31.5%
20090922	10	120	3233.5	131.1	315	0	1.3	2922	37.4%
20090922	10	300	2838.9	317.8	85	0	1.0	2736	45.0%
20090922	3	3600	2508.0	2348.3	74	0	1.0	2425	51.4%
20090923	10	120	1659.4	244.0	50	0	0	1589	17.1 %
20090923	10	300	1550.1	592.1	45	0	0	1464	22.5%
20090923	3	3600	1419.0	4420.7	34	0	0	1375	29.3%
20090924	10	120	2822.5	141.5	142	0	1.1	2670	30.6%
20090924	10	300	2578.8	309.7	85	0	1.0	2415	36.5%
20090924	3	3600	2197.3	2716.7	24	0	1.0	2163	45.9%
20090925	10	120	2051.8	150.4	60	0	0	1935	41.4%
20090925	10	300	1974.0	367.2	38	0	0	1910	43.6%
20090925	3	3600	1749.3	4252.3	13	0	0	1732	50.0%
20090926	10	120	1481.0	246.0	39	0	0	1422	16.9%
20090926	10	300	1351.9	597.3	53	0	0	1303	21.1%
20090926	3	3600	1139.0	5854.3	28	0	0	1103	36.1%
20090927	10	120	1899.4	212.7	34	0	0	1852	38.6%
20090927	10	300	1770.9	516.7	70	0	0	1672	42.8%
20090927	3	3600	1512.3	5927.3	29	0	0	1477	51.1%
20090928	10	120	1623.8	224.2	49	0	0	1533	26.5%
20090928	10	300	1512.9	544.8	46	0	0	1445	31.5%
20090928	3	3600	1329.3	5586.3	36	0	0	1281	39.8%
20090929	10	120	2221.3	123.8	85	0	0	2043	31.8%
20090929	10	300	2102.3	302.1	48	0	0	2003	35.5%
20090929	3	3600	1855.7	2826.0	47	0	0	1800	43.1%
20090930	10	120	7351.9	103.4	1478	2.8	5.0	4051	45.3%
20090930	10	300	5142.8	250.1	1264	1.3	3.3	3992	61.7%
20090930	3	3600	3524.3	2141.0	338	1.0	0.7	3247	73.9 %
20091001	10	120	2777.1	112.1	168	0	0.3	2554	36.9%
20091001	10	300	2470.0	270.0	162	0	0.1	2230	43.9%
20091001	3	3600	2062.3	1925.3	17	0	0	2038	53.1%

Table D.5: The results of simulated annealing running for two minutes, five minutes and one hour.

In Table D.5 we have for each data set tested the simulated annealing for 120 seconds (2 minutes), 300 seconds (5 minutes) and 3600 seconds (1 hour) as given

in column three. For the tests allowed 120 second and 300 seconds each test is repeated ten times. However, for the tests running 3600 seconds each test is only repeated three times as given in column two.

The temperature, cooling rate and number of moves are set to the selected values and the excess time allowed on each segment is set to $H = 30$ (minutes).

In column four the average of the solution for the runs is reported. The average number of iterations of the runs and the standard deviation of the solutions is reported in column five and six. In column seven and eight respectively the average number of unassigned prebooked PRMs and unassigned immediate PRMs are reported. The best solutions of all the runs is reported in column nine and the gap between the average solution and the initial solution from Table D.4 is reported in column ten. The gap is in Table D.5 calculated as described in Section D.6. The results in Table D.5 show that after running simulated annealing for two minutes the initial greedy heuristic solution is significantly improved. Note, that more solution time improves the solution quality in all cases and in some cases this improvement is significant. However, the improvements achieved from the initial solution in the first five minutes is in all cases greater than the improvements achieved in the following fifty five minutes. The improvement achieved in the first two minutes from the initial solution is greater in all cases except one (20090926) than the improvement achieved in the following fifty eight minutes. This indicates that the algorithm is good at finding improvements early in the algorithm and therefore works well with the requirement of solutions within a couple of minutes.

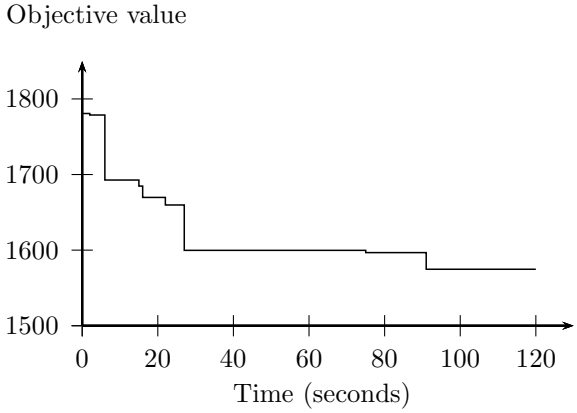


Figure D.3: The solution values over time for test case 20090926

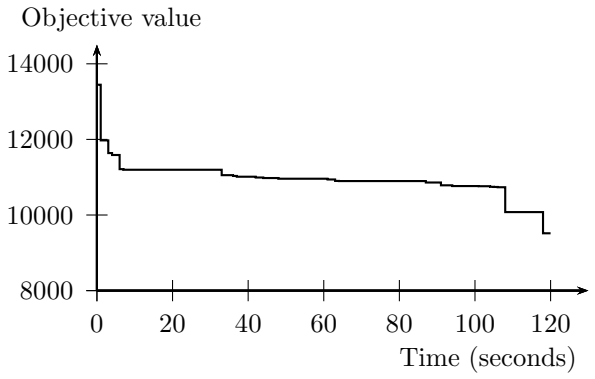


Figure D.4: The solution values over time for test case 20090930

Since the simulated annealing algorithm is tuned for runs of 2 minutes it is obvious that the acceptance of worse solutions will occur in the first 2 minutes. To show the improvement of solution value for the required solution time we have in graphs D.3 and D.4 shown the runs for only the first 2 minutes.

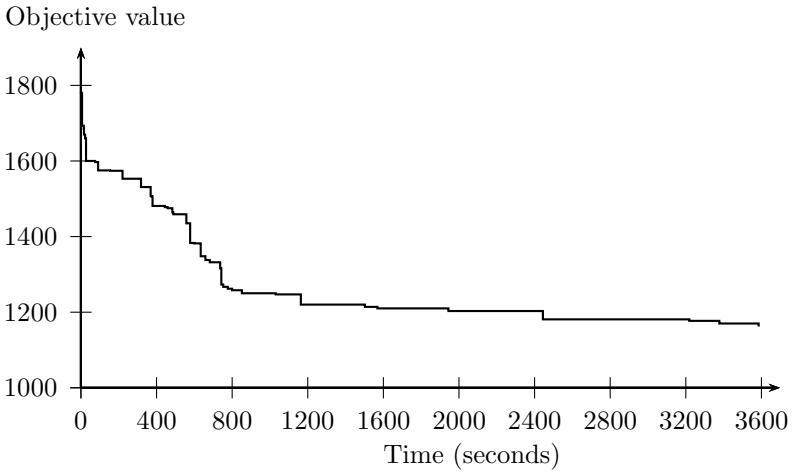


Figure D.5: The solution values over time for test case 20090926

The graphs in Figure D.5 and D.6 show the development in solution values for a single 1 hour run of respectively test instance 20090926 and 20090930. For the test instance 20090926, Figure D.5 show that after running simulated

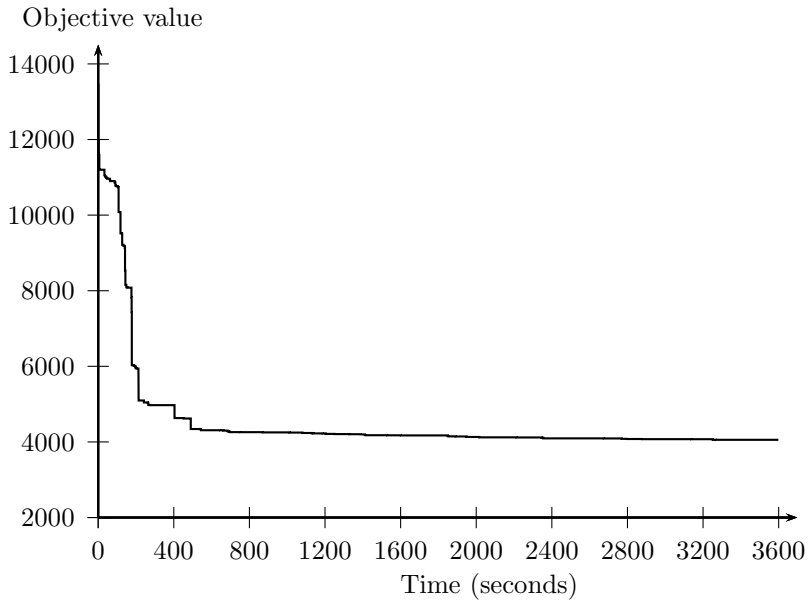


Figure D.6: The solution values over time for test case 20090930

annealing for 15 minutes the improvements to the solutions become seldom and insignificant, for the test instance 20090930 this is reduced to 10 minutes. This means that the significant reductions occurs early in the simulated annealing and very good quality solutions can be found after 10–15 minutes. In our case the solution time required is quite limited however, if significant solution times where permitted making a random restart of the algorithm with a new ordering of the lists may allow for searches in other areas of the search space. However, since the neighborhood used is very large it is easier for the algorithm to escape local minima.

D.8 Conclusion

We have presented a model for the airport PRM transport problem (APRMTP) and developed a heuristic with promising results even when the running time is two minutes as required by the service company. Moreover the number of rejected PRMs is very low also in the initial greedy insertion heuristic solutions.

Although the problem has been defined incorporation with a specific handling company, we believe that the developed model is sufficiently general to cover most airports in the world.

The program developed seems to work very well with the short solution time constraint as the big improvements are obtained within the first few minutes.

From the tests and the solution graphs it is clear that increasing the solution time a little could in some cases give significant improvements. In stead of increasing the solution time such improvements could also be achieved by increasing computational power or if possible algorithmic improvements such as parallelization of the program.

The next step in the academic sense would be to test different strategies on the problem presented and compare the solutions quality and to reduce computation time for the greedy insertion heuristic. In the application sense the next step would be to in cooperate this method at the users and to see if the use of our plan can improve their daily service.

We have assumed that the personnel and the location of the personnel is fixed, but the short solution times used by the developed algorithm makes it possible to use local search to test whether relocation of some personnel may lead to a higher service level. Since the number of passengers arriving at the airport is increasing the algorithm can be used to find when an increase in personnel is needed to deliver acceptable service to the increasing volumes of PRMs arriving.

Acknowledgments

The authors wish to thank Torben Barth and Berit Løfstedt for valuable comments. The authors also wish to thank Jacob Colding for presenting and clarifying the problem to us.

References

[1] Gatwick managing directors report, 2009.

[2] The worlds top 50 airports. *Air Transport World*, 47:51, 2010.

[3] J. W. Baugh, G. K. R. Kakivaya, and J. R. Stone. Intractability of the dial-a-ride problem and a multiobjective solution using simulated annealing. *Engineering Optimization*, 30:91–123, 1998.

[4] P. Chen, Y. Guo, A. Lim, and B. Rodrigues. Multiple crossdocks with inventory and time windows. *Computers & Operations Research*, 33:43–63, 2006.

[5] J.-F. Cordeau. A branch-and-cut algorithm for the deal-a-ride problem. *Operations Research*, 54:573–586, 2006.

[6] J.-F. Cordeau and G. Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153:29–46, 2007.

[7] T. G. Crainic, N. Ricciardi, and G. Storchi. Models for evaluating and planning city logistics systems. *Transportation Science*, 43:432–454, 2009.

[8] M. Diana and M. M. Dessouky. A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. *Transportation Research Part B*, 38:539–557, 2004.

[9] A. Dohn, M. S. Rasmussen, and J. Larsen. The vehicle routing problem with time windows and temporal dependencies. Technical report, DTU Management, The Technical University of Denmark, 2009.

[10] T. Feo and M. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[11] I. Ioachim, J. Desrosiers, F. Soumis, and N. Belanger. Fleet assignment and routing with schedule synchronization constraints. *European Journal of Operational Research*, 119:75–90, 1999.

[12] J. Jaw, A. Odoni, N. Psaraftis, and H. M. Nigel. A heuristic algorithm for the multi-vehicle advance request deal-a-ride problem with time windows. *Transportation Research part B*, 20:243–257, 1986.

[13] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435, 2007.

[14] S. Ropke, J.-F. Cordeau, and G. Laporte. Models and branch-andalgorithms for pickup and delivery problems with time windows. *Networks*, 49:258–272, 2007.

[15] M. Wen, J. Larsen, J.-F. Cordeau, and G. Laporte. Vehicle routing with cross-docking. *Journal of the Operational Research Society*, 60:1708–1718, 2009.

[16] Z. Xiang, C. Chu, and H. Chen. A fast heuristic for solving a large-scale dial-a-ride problem under complex constraints. *European Journal of Operational Research*, 174:1117–1139, 2006.

Paper E

The Offline Group Seat Reservation Problem

Tommy Clausen, Allan Nordlunde Hjorth, Morten Nielsen and David Pisinger

Published in European Journal of Operational Research, Volume 207, Issue 3, pp 1244-1253, 2010.

The Offline Group Seat Reservation Problem

Tommy Clausen, Allan Nordlunde Hjorth, Morten Nielsen and David Pisinger

In this paper we address the problem of assigning seats in a train for a group of people traveling together. We consider two variants of the problem. One is a special case of two-dimensional knapsack where we consider the train as having fixed size and the objective is to maximize the utilization of the seats in the train. The second is a special case of two-dimensional bin packing where all requests must be accommodated while trying to minimize the number of passenger cars needed. For both variants of the problem we present a number of bounds and develop exact algorithms. Computational results are presented for various instances based on realistic data, and from the packing literature adapted to the problems addressed.

E.1 Introduction

We are considering the off-line group seat reservation problem (GSRP). In this problem it is the objective to maximize the use of the seats in a train subject to a number of constraints: A train consists of a number of seats which are numbered consecutively. A seat reservation brings a person from a station a to a station b without changing seat. A group of people, all having the same station of origin and destination, may wish to sit together, i.e. being assigned to seats with consecutive numbers. In the off-line version we assume that all data are given in advance.

The GSRP can be interpreted geometrically in the following way. A (group) reservation can be represented by a rectangle having width equal to the number of seats reserved and height equal to the distance traveled. For the train, the entire route corresponds to the height and the availability of seats is represented by the width. This corresponds to a two-dimensional orthogonal packing problem where no rotation is allowed and where the position of each reservation is fixed in height.

The example in Figure E.1 illustrates the geometric interpretation of the GSRP.

A train with three seats travels four stations from y_1 to y_4. The five given reservations and the corresponding packing is illustrated in the figure. We shall in the following use the terms reservation and rectangle interchangedly.

reservation	no. seats	from	to
1	1	y_1	y_3
2	2	y_1	y_2
3	1	y_2	y_3
4	1	y_2	y_4
5	2	y_3	y_4

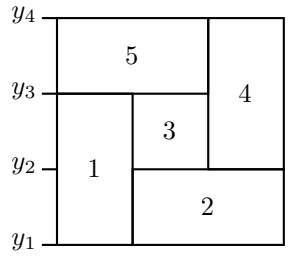


Figure E.1: The geometric interpretation of a group seat reservation problem instance. The train travels from station y_1 to station y_4. Five reservations are given in the table to the left and a packing of the train is shown to the right.

The GSRP has numerous applications. In its direct formulation it reflects transportation of children going to a school camp. All requests are known in advance, and school classes may not be split for safety reasons. A similar problem appears when considering reservation of hotel rooms, where people wish to have adjacent rooms. A storehouse may have a number of shelves of fixed length. We know in advance which item will arrive at given date and how long time it will stay in the store house. Each item has a given length, and the sum of the item lengths at each shelf may not exceed the overall shelf capacity. In the well-studied berth scheduling problem [15, 24] a number of vessels are planned to arrive to a quay. The arrival and departure time of each vessel is known as well as the length of the vessel. In the classical berth scheduling problem it is allowed to postpone the arrival of the vessels in order to accommodate all vessels. In our model, each vessel has an associated profit, and we are allowed to reject some vessels (which will choose a different quay for loading/unloading).

Finally, the GSRP appears when visualizing solutions to the finite capacity production planning problem [31]. In this problem a production plan is constructed subject to some limited resource. A solution to the problem describes when each job starts and finishes, and how much of the resource it makes use of. A graphical presentation of a solution should depict each task as a rectangle, where the x-dimension is the time, and the y-dimension is the capacity consumption. Only a fixed height is available, corresponding to the capacity in the GSRP.

In the on-line version of the GSRP, reservation requests arrive one by one, and should be assigned a group of seats immediately. Boyar and Medvedev [9] considered the single-customer version of the on-line problem, and showed that if all tickets have the same price, first-fit and best-fit are better than worst-fit in the relative worst-order ratio. This also holds for the case where the price of

the ticket is proportional to the distance traveled. Moreover, they showed that the off-line version of the single-customer GSRP where all tickets have the same price, is equivalent to a maximum k-colorable subgraph problem for interval graphs, which can be solved in polynomial time, as shown by Yannakakis and Gavril [34]. Helliesen [23] presented a new algorithm, scan-fit, for the on-line single-customer seat reservation problem, that was shown to perform better than first-fit and best-fit in the competitive ratio, both theoretically and empirically. Helliesen also considered the on-line version of the GSRP. In his version it is the objective to utilize the seats as effectively as possible while trying to minimize the distance between the people in a group. The developed algorithms also take into account how seats are arranged in the train by adding distance tables.

In this paper we describe two variants of the GSRP inspired by two-dimensional packing. Definitions and terminology follow in Section E.2. The first variant, the group seat reservation knapsack problem, is considered in Section E.3, upper bounds are derived in Section E.3.1 and an exact algorithm is presented in Section E.3.2. In addition, a method for testing if a subset of reservations can be feasibly packed within the train is presented in Section E.3.3. The second variant, the group seat reservation bin packing problem is described in Section E.4. Lower bounds for this problem are described in Section E.4.1 and an exact algorithm is presented in Section E.4.2. Finally, computational results for both variants are presented in Section E.5.

E.2 Definitions and Terminology

Using the terminology from bin packing we may assume that the train stops at H stations, and contains W seats. Let $N = \{1, \cdots, n\}$ be the set of requests, each request j asking for a number w_j of seats, traveling h_j stations from station y_j to station $y_j + h_j$. Without loss of generality we may assume that $w_j \leq W$.

Although H in principle may be large, we may reduce the problem to only consider the active stations, i.e.

$$Y := \{y_j \mid j \in N\} \cup \{y_j + h_j \mid j \in N\}$$

and let N_y be the set of requests using a seat at station $y \in Y$

$$N_y := \{j \in N \mid y_j \leq y < y_j + h_j\}.$$

We associate with each station $y \in Y$ a "height" H_y to represent the distance from station y to the next active station in Y.

By considering the train as a single row of seats, we formulate the group seat reservation knapsack problem (GSR-KP) as the problem of choosing the set of requests that maximizes the utilization of the train. We measure the utilization of the train as the number of seats times the distance traveled for all chosen reservations. This problem is a special case of the two-dimensional knapsack problem (2DKP), in which a number of rectangles have to be placed inside a larger sheet. Each rectangle has an associated profit and the objective is to place the rectangles on the sheet so as to maximize the overall profit. For the GSR-KP the profit is the area of the rectangles. In the original 2D KP, the so-called guillotine constraint may be valid. It says that the layout can be recursively separated into the individual rectangles by cuts going between two opposite edges of the current block, parallel to the other edges. This constraint makes sense for technological cutting process and will not be considered here, see Figure E.1.

The problem may be formulated as the following integer-programming model. Let $\delta_i = 1$ if request i is selected. Let x_i be the first seat (left coordinate) of request i. Let $E = \{(i, j)\}$ be the set of rectangle pairs which in some way share a station (y-coordinate). Finally, let $\ell_{ij} = 1$ iff request i is located left of j.

$$\max \quad \sum_{j \in N} w_j h_j \delta_j \tag{E.1}$$

$$\text{s.t.} \quad \sum_{j \in N_y} w_j \delta_j \leq W, \qquad y \in Y \tag{E.2}$$

$$\delta_i + \delta_j - \ell_{ij} - \ell_{ji} \leq 1 \qquad (i, j) \in E \tag{E.3}$$

$$x_i - x_j + W\ell_{ij} \leq W - w_i \qquad (i, j) \in E \tag{E.4}$$

$$0 \leq x_j \leq W - w_j \qquad j \in N \tag{E.5}$$

$$\ell_{ij} \in \{0, 1\} \qquad (i, j) \in E \tag{E.6}$$

$$\delta_j \in \{0, 1\} \qquad j \in N \tag{E.7}$$

$$x_j \geq 0 \qquad j \in N \tag{E.8}$$

Here (E.2) says that we may not exceed the capacity W of the train at any station. Constraint (E.3) says that if both requests i and j are selected, then one of them should be to the left of the other i.e. $(\delta_i = 1 \wedge \delta_j = 1) \Rightarrow (\ell_{ij} = 1 \vee \ell_{ji} = 1)$. Constraint (E.4) says that if request i is located to the left of request j then this should be reflected in the coordinates, i.e. $\ell_{ij} = 1 \Rightarrow x_i + w_i \leq x_j$. Constraint (E.5) says that all requests should be placed inside the train. Constraints (E.6) and (E.7) say that the ℓ_{ij} and δ_j variables should be binary and (E.8) say that the reservation coordinate variables x_j should be non-negative.

The problem is $\mathcal{NP}$-hard, which can easily be shown by reduction of the subset sum problem to a group seat reservation knapsack problem with no intermediate stations.

Trains usually consist of several passenger cars, rather than a single passenger compartment. This poses additional restrictions on the placement of the reservations in the train. Not only must reservations be assigned consecutive seats, they must also be assigned seats in the same car to be adjacent. This defines another problem, namely the group seat reservation bin packing problem (GSR-BPP) as the problem of assigning reservations to cars such that all requests are fulfilled and the number of cars used are minimized. The GSR-BPP is a special case of the two-dimensional bin packing problem (2DBPP), in which a number of rectangles must be assigned to identical bins, such that no bin is overfilled and the number of bins used is minimized. Regarded as a special case of 2DBPP, the train cars correspond to bins and the reservations correspond to the rectangles that must be assigned to bins. Additionally, the rectangles have fixed y-coordinates in the GSR-BPP.

More formally, we define the GSR-BPP as follows. Let W determine the number of seats in a car, and let all other variables from formulation (E.1)–(E.7) have the same interpretation in the GSR-BPP as in the GSR-KP. Furthermore, let m_i identify the car that request i is in, and let $p_{ij} = 1$ if request i is placed in a car closer to the front of the train than request j (i.e. if $m_i < m_j$). Finally v denotes the number of cars used and n is the number of requests. The problem may then be formulated as:

$$\begin{aligned}
\min \quad & v && && \text{(E.9)}\\
\text{s.t.} \quad & \ell_{ij} + \ell_{ji} + p_{ij} + p_{ji} \geq 1 && (i,j) \in E, i < j && \text{(E.10)}\\
& x_i - x_j + W\ell_{ij} \leq W - w_i && (i,j) \in E && \text{(E.11)}\\
& m_i - m_j + np_{ij} \leq n - 1 && (i,j) \in E && \text{(E.12)}\\
& 0 \leq x_j \leq W - w_j && j \in N && \text{(E.13)}\\
& 0 \leq m_j \leq v && j \in N && \text{(E.14)}\\
& \ell_{ij}, p_{ij} \in \{0,1\} && (i,j) \in E && \text{(E.15)}\\
& m_j \in \mathbb{N} && j \in N && \text{(E.16)}\\
& x_j \geq 0 && j \in N && \text{(E.17)}
\end{aligned}$$

Constraint (E.10) states that either requests i and j may not overlap, or they must be in different cars. Constraint (E.11) enforces that x-coordinates must reflect the values of the ℓ variables, i.e. $\ell_{ij} = 1 \Rightarrow x_i + w_i < x_j$. Similarly, constraint (E.12) enforces that if request i is placed in front of request j (car-wise), it must have a lower car number, i.e. $p_{ij} = 1 \Rightarrow m_i < m_j$. Constraint (E.13) states that a request must be placed entirely inside a car, and constraint (E.14) bounds the number of cars used.

The formulation (E.9)–(E.16) is $\mathcal{NP}$-hard, which can be realized by reducing from one-dimensional bin packing to the special case where all requests travel the entire train route.

To the authors' knowledge, no previous literature exist on the off-line GSR-KP or GSR-BPP. However, much related work has been done on the two-dimensional knapsack problem and the two-dimensional bin packing problem.

A number of algorithms have been presented for the 2DKP. Hadjiconstantinou and Christofides [22] studied various IP formulations, but were only able to solve instances of moderate size. Fekete and Schepers [20] presented a two-phase algorithm which first selects a subset of rectangles with large profits, and then tests feasibility through an enumerative algorithm based on isomorphic packing classes. Caprara and Monaci [10] presented an $(\frac{1}{3} - \epsilon)$-approximation algorithm for the 2DKP and developed four exact algorithms based on various enumeration schemes. Belov [5] consider a bounded 2DKP in which the placement of the rectangles should follow a two-stage guillotine packing pattern. Pisinger and Sigurd [32] used constraint programming to solve the general and guillotine-restricted version of the 2DKP. The algorithm follows a two-phase approach as in Fekete and Schepers [20] by first choosing a subset of rectangles to pack in the sheet (by solving a one-dimensional knapsack problem), and then testing whether the rectangles actually fit into the sheet. If the rectangles do not fit into the sheet a new constraint is added to the one-dimensional knapsack problem. Baldacci and Boschetti [3] and Clautiaux et al. [14] followed the same two-phase approach but used a different technique for testing feasibility of the orthogonal packing problem. Baldacci and Boschetti [3] presented a cutting-plane approach using a number of knapsack, dominance and incompatibility constraints. Clautiaux et al. [14] modeled the problem as a scheduling problem making it possible to use powerful constraint-based scheduling propagation techniques.

Berkey and Wang [6] presented an extensive computational review of heuristics for finding upper bounds for two-dimensional bin packing problems (2DBPP). The heuristics considered were mostly similar to well-known on-line algorithms like best-fit or next-fit. For a thorough presentation of on-line packing algorithms we refer to Csirik and Woeginger [16].

A number of lower bounds for the 2DBPP exist, mostly based on adaptations of bounds for one-dimensional bin packing (see e.g. Martello and Vigo [27]). More recent bounds include Fekete and Schepers [18] and Boschetti and Mingozzi [7], [8]. The latter also present a heuristic based upper bound. Clautiaux et al. [13], give a survey of dual feasible functions which are used for finding lower bounds to various cutting and packing problems.

Martello and Vigo [27] present an exact algorithm for the 2DBPP. The algorithm consists of an outer branch-decision tree in which rectangles are assigned to bins. At every node a heuristic is used to find a feasible solution for the assignment. If none is found an inner branch-decision tree is created to test the feasibility of the assignment. An additional heuristic is used to close bins if no unassigned

rectangles can fit into the bin. A similar approach was used by Martello, Pisinger and Vigo [28] to solve the three-dimensional bin packing problem.

Finally, Lodi, Martello and Vigo [26] provide a detailed survey of bounds, exact algorithms and heuristics for the 2DBPP.

E.3 The Group Seat Reservation Knapsack Problem

As mentioned in the introduction we consider the group seat reservation knapsack problem where the train is considered as having all seats in a single row. In this section we present a number of upper bounds that we use to develop an exact algorithm for the problem.

E.3.1 Upper Bounds

A first upper bound may be obtained by LP-relaxing the integer programming model for GSR-KP defined by (E.1)–(E.8). The optimal solution to this model gives us the upper bound $\mathcal{U}_1$. Notice that for any LP-optimal solution to (E.1)–(E.8) we may get an equivalent LP-solution by setting $x_i := 0$ and $\ell_{ij} := (W - w_i)/W$ for any i and j. In this way constraints (E.4) and (E.5) are always satisfied when (E.2) are trivially satisfied. Moreover (E.3) is satisfied, since with the above definition of ℓ_{ij} it reads

$$\delta_i + \frac{w_i}{W} + \delta_j + \frac{w_j}{W} \leq 3$$

Using Lemma 1 from Appendix A gives the stated. What remains is the problem

$$\begin{aligned} \max \quad & \sum_{j \in N} w_j h_j \delta_j \\ \text{s.t.} \quad & \sum_{j \in N_y} w_j \delta_j \leq W, \quad y \in Y \\ & 0 \leq \delta_j \leq 1 \qquad\quad j \in N \end{aligned} \tag{E.18}$$

Now, suppose we introduce a variable δ_j^k for each single seat, such that $w_j\delta_j = \sum_{k=1}^{w_j} \delta_j^k$. This yields the formulation

$$\begin{aligned} \max \quad & \sum_{j\in N}\sum_{k=1}^{w_j} h_j\delta_j^k \\ \text{s.t.} \quad & \sum_{j\in N_y}\sum_{k=1}^{w_j} \delta_j^k \leq W, \quad y \in Y \\ & 0 \leq \delta_j^k \leq 1 \qquad\qquad j \in N,\ k = \{1,\cdots,w_j\}, \end{aligned} \tag{E.19}$$

which is equivalent to (E.18).

Let A be the constraint matrix of (E.19). In each column of A the ones will appear consecutively, since the reservations are connected. This is known as the "consecutive ones" property which implies that A is totally unimodular. Consequently, an optimal solution to (E.19) exists in which $\delta_j^k \in \{0,1\}$. This means that (E.19), and thereby (E.18), is equivalent to splitting the reservations into single-seat reservations.

Denote by G the intersection graph of the columns of A, i.e. G contains a vertex for each column in A and an edge between two vertices if there exists a row in A that contains ones in both the corresponding columns. Equivalently, G contains a vertex for each single-seat reservation, and two vertices in G are connected by an edge if the corresponding reservations share a station. By considering the reservations as intervals on the time axis, we note that G is an interval graph (and is thereby perfect).

If we assign the reservation travel distances as weights to the vertices of G, we may formulate (E.19) as a weighted W-independent set problem. The weighted W-independent set problem considers a graph with a non-negative weight associated with each vertex. The problem is then to find W independent sets such that the weight sum of all vertices in the independent sets is maximized. The problem is $\mathcal{NP}$-hard in general, but can be solved polynomially if the graph is an interval graph (see e.g. [30]). This graph interpretation of the single-seat reservation problem is also noted in [9], although they noted it for the also equivalent W-colorable subgraph problem.

For the weighted W-independent set problem on interval graphs, Pal and Bhattacharjee [30] present an $O(Wm\sqrt{\log c} + \gamma)$ time algorithm, where m is the number of vertices in the interval graph, c is the weight of the longest path in the graph and γ is the total size of all maximal cliques in the graph. Using this algorithm to solve (E.18), we can rewrite the running time using the notation of the GSR-KP. The number of independent sets is then the number of seats in

the train W. The number of vertices m is the total number of seats in all reservations, i.e. $m = \sum_{j=1}^{n} w_j$. The weight of the longest path c is bounded above by the total travel length of all reservations, i.e. $c \leq \sum_{j=1}^{n} h_j$. The number of cliques in an interval graph is at most the number of vertices (see e.g. [21]), so γ is bounded by $m^2 = \left(\sum_{j=1}^{n} w_j\right)^2$. Thus, the complexity is

$$O\left(W \cdot \sum_{j=1}^{n} w_j \cdot \sqrt{\log \sum_{j=1}^{n} h_j} + \left(\sum_{j=1}^{n} w_j\right)^2\right). \tag{E.20}$$

By noting that $\sum_{j=1}^{n} w_j \leq N \cdot W$ and $\sum_{j=1}^{n} h_j \leq N \cdot H$ expression (E.20) can be reduced to

$$O\left(NW^2\sqrt{\log(NH)} + (NW)^2\right)$$

which is clearly pseudo-polynomial in terms of the GSR-KP.

We have described two methods for calculating the single-seat relaxation. We shall denote by $\mathcal{U}_1$ the bound calculated by solving the LP-relaxation of (E.1) – (E.7) and denote by $\mathcal{U}_2$ the bound calculated by the weighted W-colorable subgraph problem as described above. Although the bounds are identical, the time complexities of calculating them are different and no dominance exists between the time bounds.

A third upper bound is obtained by relaxing the problem to the case where the passengers may need to change seats at every station.

$$\begin{aligned} \max \quad & \sum_{j \in N} w_j h_j \delta_j & & \\ \text{s.t.} \quad & \sum_{j \in N_y} w_j \delta_j \leq W & y \in Y & \\ & \delta_j \in \{0,1\} & j \in N & \end{aligned} \tag{E.21}$$

The above problem is a multidimensional knapsack problem with $|Y|$ knapsack constraints, which is strongly $\mathcal{NP}$-hard to solve [25]. Let us denote the optimal value of (E.21) by $\mathcal{U}_3$. Clearly, $\mathcal{U}_1$ and $\mathcal{U}_2$ are the LP-relaxation of $\mathcal{U}_3$, and hence are dominated by $\mathcal{U}_3$.

A fourth upper bound may be found as follows: For every station (y-coordinate), we calculate how well the train may be filled at this station, by solving the following subset sum problem:

$$F_y = \max\left\{ \sum_{j \in N_y} w_j \delta_j \,\middle|\, \sum_{j \in N_y} w_j \delta_j \leq W, \delta_j \in \{0,1\} \right\}.$$

We may now calculate an upper bound as

$$\mathcal{U}_4 = \sum_{y \in Y} F_y H_y.$$

where H_y is the "height" of station y as defined in the beginning of Section E.2. Calculating $\mathcal{U}_4$ is weakly $\mathcal{NP}$-hard as it contains $|Y|$ subset sum problems.

A similar subset sum problem is used by Clautiaux et al. [14] to prune the search tree by using subset sum problems to determine feasibility.

Bound Dominance

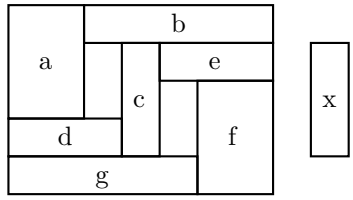
The bound $\mathcal{U}_4$ is calculated by splitting the reservations into smaller reservations each traveling only one station. In the interpretation of allowing seat changes, this corresponds to allowing seat changes to "outside the train", i.e. passengers are allowed to leave the train and join it again at a later station. This is clearly less restricted than $\mathcal{U}_3$ in which seat changes must be to other seats within the train. Thus, $\mathcal{U}_3$ dominates $\mathcal{U}_4$.
That $\mathcal{U}_3$ dominates $\mathcal{U}_1$ and $\mathcal{U}_2$ is seen by comparing formulations (E.18) and (E.21). Clearly, $\mathcal{U}_1$ and $\mathcal{U}_2$ are the LP-relaxation of $\mathcal{U}_3$.
The bounds $\mathcal{U}_1$ and $\mathcal{U}_4$ split reservations in different ways: $\mathcal{U}_1$ splits into single-seat reservations and $\mathcal{U}_4$ splits into single-station reservations. Thus, $\mathcal{U}_1$ and $\mathcal{U}_4$ do not dominate each other. But since the problem is already restricted with regard to stations, we may expect $\mathcal{U}_1$ to be the tighter of the two bounds in general.

Theorem 1 *The bound $\mathcal{U}_1$ does not dominate $\mathcal{U}_4$ and $\mathcal{U}_4$ does not dominate $\mathcal{U}_1$.*

Proof. Consider a train with height H and width W and two reservations r_1 and r_2 which both have height H (i.e. they travel from the first to the last station) and require $\lfloor W/2 \rfloor + 1$ seats. Clearly, both r_1 and r_2 cannot be assigned seats in the train. By splitting the reservations into single-seat reservations, the entire train can be filled, so $\mathcal{U}_1 = H \cdot W$. If the reservations are split into single-station reservations, each station can still only accommodate r_1 or r_2. Thus, $\mathcal{U}_4 = H \cdot (\lfloor W/2 \rfloor + 1)$.
Conversely, consider the reservations s_1 and s_2 that both request W seats. Let s_1 travel from station 1 to station $\lfloor H/2 \rfloor + 1$ and s_2 from station $\lfloor H/2 \rfloor$ to station H. Splitting s_1 and s_2 into single-seat reservations will not change that station $\lfloor H/2 \rfloor$ can accommodate only W seats. The W largest single-seat reservations will originate from s_1, so $\mathcal{U}_1 = W \cdot (\lfloor H/2 \rfloor + 1)$. If the reservations are split into single-station reservations, only station $\lfloor H/2 \rfloor$ will be overfilled. This, and all

other stations can be filled completely, so $\mathcal{U}_4 = W \cdot H$. □

As $\mathcal{U}_3$ dominates all the other bounds and is itself $\mathcal{NP}$-hard, it may be that $\mathcal{U}_3$ is simply a more simple formulation of the GSR-KP. This is not the case as is shown by the following example with 7 seats, reservations a–g, and x as illustrated below. The packing of a-g represents an optimal solution. The bound $\mathcal{U}_3$ will split x into single-station reservations which may be placed next to c. Thus, $OPT < \mathcal{U}_3$ for this instance.



E.3.2 Exact Algorithm

In Section E.3.1 we have used different ideas to develop upper bounds for the GSR-KP. For all the bounds it is possible that the calculated value is optimal for the GSR-KP, but it is more likely to be larger than optimum. Since we wish to develop an exact algorithm for solving the GSR-KP we use branch and bound.

In each node of the branching tree we choose a rectangle j and divide the solution space into two subtrees. In one subtree we demand that the chosen rectangle is in the packing and in the other subtree, we exclude the rectangle from the packing. In the integer programming model (E.1)–(E.7) this corresponds to branching on the δ variables fixing δ_j to 1 respectively 0. We thereby get two new subproblems. The first subproblem is equivalent to saying that the width of the train in reduced by w_j on all stations covered by the rectangle. The second subproblem corresponds to removing the chosen rectangle from the set of rectangles that should be packed.

We start by fixing the largest rectangles. This way we will very early in the branching tree get to a point where there is no more room for extra rectangles and we can prune large parts of the branching tree.

When we fix a rectangle we check each station separately to see if there is enough

room for the rectangle, but this does not ensure that there exist a legal packing of the fixed rectangles. Therefore at some point we need to test for feasibility i.e. find out if there exist a legal packing of the chosen rectangles. One can consider different schemes for testing feasibility. One possibility is to test for feasibility at each node, but since we need to find a packing to ensure that it is legal, this scheme requires solving an $\mathcal{NP}$-complete problem at each node, thus we choose to only test for feasibility when all rectangles have been fixed. How the actual testing of feasibility is done, is described in Sections E.3.3–E.3.5.

Since it is very important that we quickly find some feasible solutions in order to prune parts of the branching tree we use a depth first strategy in our branching.

E.3.3 Testing Feasibility

When testing the feasibility of a packing, we are given a set of reservations and we then wish to determine if the reservations can be placed in a way to make them fit into the train. Placing the reservations within the train corresponds to assigning values to the binary ℓ-variables of equations (E.3) and (E.10) for the GSR-KP and GSR-BPP, respectively. Recall that $\ell_{ij} = 1$ means that reservation i is positioned to the left of j.
The test for feasibility consists of two parts. An algorithm that determines the feasibility of a single packing (i.e. assignment of the ℓ-variables) is described in section E.3.4 and an enumeration scheme for applying this algorithm to every assignment of ℓ_{ij} variables is described in section E.3.5.

E.3.4 Feasibility of a Packing

We shall first consider the feasibility of a packing where the relative positions of the rectangles are known, i.e. the left-right ordering of overlapping rectangles is predetermined. To test the feasibility of the packing, we will use a graph representation by Fekete and Schepers [19], and validation techniques by Pisinger and Sigurd [32]. The reason for choosing the framework by Fekete and Schepers [19] is that the additional constraints imposed by our problem give rise to some nice graph theoretical properties in the representation. However, instead of using interval graphs, whose complements are orientable, as in [19], we work already on directed graphs.

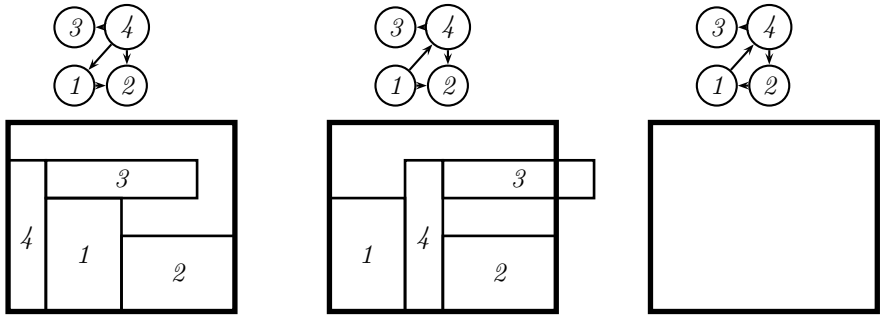
By exploiting that the rectangles are fixed in the y-dimension, we may represent a packing by using a graph as follows: Let $G = (V, E)$ be a directed graph with a vertex for each reservation and the edge $(i, j) \in E$ iff $\ell_{ij} = 1$. For ease of notation

we shall not distinguish between a vertex and its corresponding rectangle. The following properties are necessary and sufficient for determining if the packing represented by G is feasible. Without loss of generality we will assume that all rectangles are placed as far to the left as possible.

P1 G is acyclic.

P2 For every path $p = \langle v_1, \ldots, v_n \rangle$ in G, $\sum_{i \in p} w_i \leq W$.

Example 1 *Consider the instance given by* $N = \{(2,3,0),(3,2,0),(4,1,3),(1,4,0)\}$, *where* $i = (w_i, h_i, y_i)$, $i \in N$ *and* $H = 6$, $W = 6$. *Below are shown three different graph representations and their corresponding packing (if legal).*



The leftmost graph represents a feasible packing with 1 *to the left of* 2 *and* 4 *to the left of all other rectangles. The middle graph contains the path* $p = \langle 1, 4, 3 \rangle$ *with* $\sum_{i \in p} w_i = 7$, *so this is infeasible by P2. The rightmost graph contains a cycle and does not represent a legal packing by P1.*

That the properties P1 and P2 are necessary follows from the middle and right graph of Example 1. That the properties are also sufficient can be seen by the following theorem, which is a special case of Theorem 1 [19]. A direct proof can be found in Appendix A.

Theorem 2 *The properties P1 and P2 are sufficient for describing a feasible packing*

The following algorithm determines if a graph satisfies the properties P1 and P2 by computing the x_j values for all reservations $j \in N$. By the assumption of leftmost placement we must set

$$x_j = \begin{cases} \max\{x_i + w_i \mid (i,j) \in E\}, & \exists i \in N : (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{E.22}$$

These values are easily calculated by a longest-path algorithm [1] which starts by topologically sorting G (and hence checking for cycles) and then running a label correcting algorithm. The time complexity is $\Theta(V + E)$. If for some j we have $x_j + w_j > W$, the packing is infeasible by P2.

E.3.5 Feasibility Enumeration Scheme

In the previous section we described how to test feasibility of a packing, given the individual placements of the reservations. To decide if a feasible packing exists for a set of reservations, all such placements must be considered. By the assumption of left-most placement (E.22) we consider the x_j variables to be uniquely determined by the corresponding ℓ_{ij} assignments. Since each placement is represented by a specific orientation of all edges in the graph, there exist an exponential number of placements. We consider all placements by using a branch-decision tree to enumerate the edge orientations in the graph. At each node in the tree we add an edge to the graph and branch on the orientation of the edge. If the graph at some node describes an infeasible packing, that subtree is eliminated since adding more edges to the graph will not make the packing feasible. If a feasible packing is found at a leaf node, the algorithm returns **true**. If no more nodes exists, the algorithm returns **false**.

E.4 The Group Seat Reservation Bin Packing Problem

So far, the train has been considered as having a single line of seats. However, most trains will consist of several cars or compartments, and it would seem unreasonable to split a group of passengers traveling together into different cars, even if their seat numbers are consecutive. Thus we now consider the group seat reservation bin packing problem. For this problem we will, as for the GSR-KP, develop an exact algorithm, but first we present some lower bounds we can use in the algorithm.

E.4.1 Lower Bounds

A first lower bound $\mathcal{L}_1$ may be found by solving the LP-relaxation of (E.9)–(E.16). It is easy to show (see [12] for details) that $\mathcal{L}_1 = 0$ will hold for any instance.

This means that $\mathcal{L}_1$ will always say that the reservations can be packing using minimum one car.

A second lower bound $\mathcal{L}_2$ may be found as follows: For every station (y-coordinate), we calculate a lower bound on the number of cars at this station. Let the binary variable $x_{ij} = 1$ iff request j is assigned to car i and let B denote the set of cars. Moreover let $\delta_i = 1$ iff car i is used.

$$\min \quad \xi_y^2 = \sum_{i=1}^{n} \delta_i \tag{E.23}$$

$$\text{s.t.} \quad \sum_{j \in N_y} w_j x_{ij} \leq W \delta_i \quad i \in B \tag{E.24}$$

$$\sum_{i \in B} x_{ij} = 1 \qquad j \in N_y \tag{E.25}$$

$$\delta_i \in \{0,1\} \qquad i \in B \tag{E.26}$$

$$x_{ij} \in \{0,1\} \qquad i \in B,\ j \in N_y \tag{E.27}$$

We may now calculate the lower bound as

$$\mathcal{L}_2 = \max_{y \in Y} \xi_y^2$$

The model (E.23)–(E.27) is recognized as an ordinary bin packing problem (BPP) which is $\mathcal{NP}$-hard to solve. Hence, to find a polynomial bound for the GSR-BPP we may use any lower bound from the literature of BPP. For each $y \in Y$ let ξ_y^3 be such a lower bound chosen as maximum of the bounds presented by Martello and Toth [29], Dell'Amico and Martello [17].

$$\begin{aligned} \xi_y^3 &= \left| \left\{ j \in N_y : w_j > \frac{W}{2} \right\} \right| \\ &+ \max_{1 \leq p \leq \frac{W}{2}} \left\{ \left\lceil \frac{\sum_{j \in N_s(p)} w_j - (|N_\ell(p)|W - \sum_{j \in N_\ell(p)} w_j)}{W} \right\rceil, \right. \\ &\qquad \left. \left\lceil \frac{|N_s(p)| - \sum_{j \in N_\ell(p)} \lfloor \frac{W - w_j}{p} \rfloor}{\lfloor \frac{W}{p} \rfloor} \right\rceil \right\} \end{aligned} \tag{E.28}$$

where

$$N_\ell(p) = \{ j \in N_y : W - p \geq w_j > \tfrac{W}{2} \} \tag{E.29}$$

$$N_s(p) = \{ j \in N_y : \tfrac{W}{2} \geq w_j \geq p \} \tag{E.30}$$

This leads to the third lower bound

$$\mathcal{L}_3 = \max_{y \in Y} \xi_y^3$$

which according to [17] can be calculated in $O(N_y)$ time for each $y \in Y$ leading to an overall time complexity of $O(N^2)$.

Finally, we expand $\mathcal{L}_2$ to consider two stations $y, y' \in Y$ at the same time. This results in a new measure $\xi^4_{y,y'}$ which extends (E.23)–(E.27). Let $x_{ij} = 1$ iff request $j \in N_y$ is assigned to car i at station y and $x'_{ij} = 1$ iff request $j \in N_{y'}$ is assigned to car i at station y'. As before let $\delta_i = 1$ iff car i is used and let B denote the set of cars.

$$\begin{aligned}
\min \quad & \xi^4_{y,y'} = \textstyle\sum_{i=1}^{n} \delta_i && && \text{(E.31)}\\
\text{s.t.} \quad & \textstyle\sum_{j \in N_y} w_j x_{ij} \le W \delta_i && i \in B && \text{(E.32)}\\
& \textstyle\sum_{j \in N_{y'}} w_j x'_{ij} \le W \delta_i && i \in B && \text{(E.33)}\\
& x_{ij} = x'_{ij} && j \in N_y \cap N_{y'} && \text{(E.34)}\\
& \textstyle\sum_{i \in B} x_{ij} = 1 && j \in N_y && \text{(E.35)}\\
& \delta_i \in \{0,1\} && i \in B && \text{(E.36)}\\
& x_{ij}, x'_{ij} \in \{0,1\} && i \in B,\ j \in N_y && \text{(E.37)}
\end{aligned}$$

Here constraint (E.32) and (E.33) are the ordinary bin packing constraints for each of the stations y, y' while (E.34) demands that a request is assigned to the same car at both stations. Constraint (E.35) enforces that each request $j \in N_y$ must be assigned a car.

We may now calculate the lower bound as

$$\mathcal{L}_4 = \max_{y,y' \in Y} \xi^4_{y,y'}$$

Calculating ξ^4 is $\mathcal{NP}$-hard, which is seen by reduction from bin packing to the special case $y = y'$. Therefore calculating $\mathcal{L}_4$ is also $\mathcal{NP}$-hard.

E.4.2 Exact Algorithm

For solving the group seat reservation bin packing problem, we construct a two-phase branching algorithm as proposed by Martello and Vigo [27].

The first phase is an outer branch-decision tree that assigns rectangles to bins, considering rectangles ordered by decreasing size. At each node the next unassigned rectangle is assigned to each of the open bins and to one new bin. A bin is *open* if at least one rectangle has been assigned to it. If the number of open bins exceeds the upper bound we may backtrack. Initially the upper bound is set as the number of rectangles, and is updated when a better feasible solution

is found. The tree is traversed in a depth first manner, that always choses the lowest numbered bin first. This delays the opening of new bins, and thus postpones the parts of the solution space using a large number of bins to a point where they may hopefully be pruned.

The second phase is run at each node of the outer tree to test the feasibility of the assigned rectangles. We use the branch-decision tree described in Section E.3.3 to test the feasibility.

Closing Bins

In addition to the two-phase branching scheme we attempt at each node to close one or more bins. If it can be determined that for some bin i, none of the unassigned rectangles fit into the bin, we mark the bin as closed. In the subtree rooted at that node, rectangles are not assigned to the closed bin.

In order to avoid creating a feasibility branch-decision tree for each unassigned rectangle in combination with each node, the following method is employed instead. For each station (y-coordinate) of the bin, the width of all rectangles that cover that station is added. If the sum exceeds the width of the bin, the packing is infeasible. Since the method is heuristic in nature, it may occur that a bin is kept open even though no feasible packing may be obtained from adding any of the remaining unassigned rectangles. This method is identical to the test performed when fixing a rectangle in the branch and bound tree for the GSR-KP (see Section E.3.2).

E.5 Computational Results

We have implemented bounds and exact algorithms for the GSR-KP and the GSR-BPP as described in Sections E.3 and E.4. The algorithms and bounds have been implemented in C and C++ using LEDA 4.5 [2]. All tests have been performed on an Intel Pentium 4 with 2 GB of memory running at 3GHz. For all tests we have limited the time consumption to 30 minutes. Tests not completed within this time limit are terminated with no result.

We start this section by looking at the upper bounds for the GSR-KP followed by the exact algorithm for the GSR-KP. Then we look at lower bounds for GSR-BPP and the performance of the exact algorithm for this problem. All computational times in the tables are in seconds.

E.5.1 Upper bounds for GSR-KP

Since we have no knowledge of any prior work on this problem new test instances for testing the algorithms have been created. We have created two types of instances. One type was inspired by existing instances taken from the literature of 2D packing. The other type was inspired by real-life problems. All instances can be downloaded from the last authors home page [1].

The packing instances were created by modifying the 2D knapsack packing instances also considered by Caprara and Monaci [10]. A detailed description of the `CGCUT` instances can be found in [11]. The `GCUT` instances are described in [4] and the `OKP` instances are described in [20]. Finally the `WANG` instances are described in [33].

We needed to add a starting station for all the reservations, while making sure that the ending stations did not exceed the final station. The `CGCUT`, `OKP` and `WANG` instances allowed several reservations of the same type. These reservations were split up into individual reservations and assigned separate starting stations. The starting stations are generated randomly and we therefore generate five new instances from each of the original instances to make sure we get some variation.

The real-life instances were created by considering the normal usage pattern of regional train services in Denmark. Most train routes cover a mix of major cities and less populated areas. Typically, the utilization of the train is high at the city stations and decreasingly lower further from the city. At peak hours, the train will not be able to accomodate all reservations around the city stations, but is unlikely to be filled outside the city areas. The generated instances reflect this by having the requests grouped near a city in the middle of the route, and almost none near the route endpoints. A series of instances with a varying number of requests and group sizes were generated.

To evaluate the quality of $\mathcal{U}_1$ we used CPLEX 9.1 to solve the problem as a linear programming problem. $\mathcal{U}_3$ is a multiple knapsack problem with no further constraints and can relatively easily be solved as an integer programming problem. Thus this bound is also solved using CPLEX. We have implemented the upper bounds $\mathcal{U}_2$ and $\mathcal{U}_4$ described in Section E.3.1.

Table E.1 gives a summary of the overall quality of the bounds on the instances derived from packing. More detailed results can be found in [12]. $\mathcal{U}_3$ is clearly the tightest of the considered bounds, as it finds the optimal solution for all of the instances. The two bounds $\mathcal{U}_1$ and $\mathcal{U}_2$ solve the same relaxation by two

[1] http://www.diku.dk/hjemmesider/ansatte/pisinger/seatres_instances.zip

different approaches, hence the bound values are the same. However, $\mathcal{U}_1$ is generally faster to calculate than $\mathcal{U}_2$.

Bound		$\mathcal{U}_1$	$\mathcal{U}_2$	$\mathcal{U}_3$	$\mathcal{U}_4$
Time	Average	0.13	0.83	0.05	0.00
	Std. dev.	0.11	1.81	0.04	0.01
Gap	Average	10.92	10.92	0.00	19.23
	Std. dev.	7.72	7.72	0.00	8.32

Table E.1: Overall comparison of the four bounds for GSR-KP. Average time (in seconds) and average gap in percent to the optimal solution and the standard deviations for the the four upper bounds.

As $\mathcal{U}_3$ is the tightest bound considered and the corresponding solution times are reasonable we will choose this bound as one of the bounds for the exact algorithm. It is also obvious that $\mathcal{U}_4$ is significantly faster to calculate so we choose this upper bound as the bound in a second version of the exact algorithm.

Due to the very high running times of $\mathcal{U}_2$ on the large instances we shall not consider it further, but use $\mathcal{U}_1$ for comparisons with the remaining bounds.

E.5.2 Results from the GSR-KP

The results from the exact algorithm for the GSR-KP are summarized in Tables E.2 and E.3. The algorithm is run twice for each instance, using the bounds $\mathcal{U}_3$ and $\mathcal{U}_4$. The results are compared to those of the CPLEX IP-solver. The running time has been limited to 30 minutes for each run.

For the packing instances, all algorithms could solve the instances within the time limit. Overall, the CPLEX solver performs slightly better than the algorithm using $\mathcal{U}_3$. However, the algorithm using $\mathcal{U}_3$ shows only a slight increase in computational time as the number of requests increase, whereas the CPLEX solver appears to be less stable in this regard.

The algorithm using $\mathcal{U}_4$ also shows good computational times for the smaller instances, but degrades significantly for the larger instances. This is because $\mathcal{U}_4$ is not very tight, so the solution space becomes too large, even though the bound is faster to compute.

The instances inspired by real-life problems are more difficult to solve, and seem to confirm the tendencies seen for the packing instances. Indeed, only the algorithm using $\mathcal{U}_3$ could solve all instances within the 30 minute time limit.

Instance	Stations	Seats	N	CPLEX	$\mathcal{U}_3$	$\mathcal{U}_4$
CGCUT01	15	10	16	0.02(5)	0.01(5)	**0(5)**
CGCUT02	40	70	23	0.79(5)	**0.14(5)**	0.49(5)
CGCUT03	40	70	62	**1.99(5)**	2.09(5)	4.35(5)
GCUT01	250	250	10	**0(5)**	0.02(5)	0.01(5)
GCUT02	250	250	20	**0.04(5)**	0.13(5)	0.15(5)
GCUT03	250	250	30	**0.08(5)**	0.41(5)	0.59(5)
GCUT04	250	250	50	**0.41(5)**	1.73(5)	4.74(5)
GCUT05	500	500	10	**0.02(5)**	0.05(5)	0.05(5)
GCUT06	500	500	20	**0.04(5)**	0.28(5)	0.39(5)
GCUT07	500	500	30	**0.06(5)**	0.64(5)	0.79(5)
GCUT08	500	500	50	**0.4(5)**	3.06(5)	10.89(5)
GCUT09	1000	1000	10	**0.02(5)**	0.05(5)	0.08(5)
GCUT10	1000	1000	20	**0.06(5)**	0.3(5)	0.7(5)
GCUT11	1000	1000	30	**0.2(5)**	1.81(5)	3.49(5)
GCUT12	1000	1000	50	**0.33(5)**	4.69(5)	13.63(5)
GCUT13	3000	3000	32	**1.09(5)**	2.9(5)	74.31(5)
OKP01	100	100	50	6.11(5)	**1.16(5)**	4.27(5)
OKP02	100	100	30	0.31(5)	0.22(5)	**0.21(5)**
OKP03	100	100	30	**0.17(5)**	0.25(5)	0.24(5)
OKP04	100	100	61	5.62(5)	**1.9(5)**	6.7(5)
OKP05	100	100	97	9.42(5)	**6.59(5)**	623.79(5)
WANG20	70	40	42	**0.07(5)**	0.4(5)	0.3(5)
Average	484.77	485.91	35.14	1.24(110)	1.31(110)	34.1(110)

Table E.2: Comparison between the CPLEX IP-solver and the exact algorithm for the GSR-KP for the packing-inspired instances. Reported is the instance name, the train size, and number of reservations N, as well as average time usage (and number of instances solved within 30 mins in parentheses) for each algorithm. The best result is shown in bold face.

Interestingly, the algorithm using $\mathcal{U}_4$ was also able to solve more instances than CPLEX.

E.5.3 Lower bounds for the GSR-BPP

The test instances used in the GSR-BPP is a further modification of the instances used in the GSR-KP. The maximum number of available seats in the train is ignored, and instead we introduce W, the number of seats in a train car. The train car sizes chosen are 10, 20 and 40. To make the new instances valid we adjust the number of seats asked for by a request. To ensure $0 < w_j \leq W$

Instance	Stations	Seats	N	CPLEX	$\mathcal{U}_3$	$\mathcal{U}_4$
G20N10_30	100	100	20	0.05(5)	**0.02(5)**	0.04(5)
G20N20_20	100	100	20	0.35(5)	**0.04(5)**	0.14(5)
G20N30_10	100	100	20	47.09(5)	**0.31(5)**	1.77(5)
G20U20_20	100	100	20	1.41(5)	**0.01(5)**	**0.01(5)**
G30N10_30	100	100	30	0.27(5)	**0.08(5)**	0.32(5)
G30N20_20	100	100	30	8.77(5)	**0.42(5)**	3.06(5)
G30N30_10	100	100	30	11.98(4)	**17.95(5)**	97.1(5)
G30U20_20	100	100	30	12.28(5)	**0.46(5)**	5.56(5)
G40N10_30	100	100	40	0.61(5)	**0.21(5)**	2.16(5)
G40N20_20	100	100	40	15.36(5)	**0.94(5)**	29.45(5)
G40N30_10	100	100	40	-	**23.77(5)**	770.97(5)
G40U20_20	100	100	40	15.34(4)	**10.33(5)**	77.06(5)
G50N10_30	100	100	50	2.9(5)	**0.85(5)**	19.89(5)
G50N20_20	100	100	50	104.48(5)	**1.98(5)**	64.81(5)
G50N30_10	100	100	50	1114.53(2)	**16.77(5)**	-
G50U20_20	100	100	50	318.15(2)	**12.67(5)**	722.35(1)
Average	100	100	35	110.24(67)	5.43(80)	119.65(71)

Table E.3: Comparison between the CPLEX IP-solver and the exact algorithm for the GSR-KP for the real-life-inspired instances. Reported is the instance name, the train size, and number of reservations N, as well as average time usage (and number of instances solved within 30 mins in parentheses) for each algorithm. A dash indicates that no instances were solved. The best result is shown in bold face.

we set

$$w_j = \begin{cases} W & \text{if } w_j \bmod W = 0, \\ w_j \bmod W & \text{otherwise.} \end{cases}$$

To avoid reservations for 0 seats, we set $w_j = W$ when $w_j \bmod W = 0$. We saw in section E.4.1 that $\mathcal{L}_1 = 0$ for all instances, so this was not implemented. $\mathcal{L}_2$ is an ordinary bin packing problem and hence it will be $\mathcal{NP}$-hard to compute this bound. $\mathcal{L}_3$ is a lower bound of $\mathcal{L}_2$ which is polynomially solvable. It is quickly computed and is generally expected to produce very tight lower bounds (see e.g. [29]). $\mathcal{L}_4$ is similar to $\mathcal{L}_2$ with some additional constraints and hence still $\mathcal{NP}$-hard to solve. This led us to suspect that $\mathcal{L}_4$ would be somewhat slower than $\mathcal{L}_2$ but possibly give tighter bounds.

In a branch and bound algorithm the time required to calculate a bound is very important. $\mathcal{L}_3$ is expected to be considerably faster than the other two and is at the same time believed to give quite good bounds. Consequently only $\mathcal{L}_3$ was chosen as the lower bound for the exact algorithm.

Using $\mathcal{L}_3$ we are able to solve the bound in only a few milliseconds. Moreover, in the cases where we have an optimal solution, $\mathcal{L}_3$ matched the upper bound in every packing instance except one (`GCUT10_1` - bin size 40), indicating that $\mathcal{L}_3$ is very tight for the considered instances.

E.5.4 Results from the GSR-BPP

Since we had no previous results to compare to, we used the CPLEX IP-solver as a reference solution. CPLEX had some difficulties solving the test instances. In fact the CPLEX IP-solver was only able to solve between 18 and 20 (depending on bin size) of the 110 packing instances in less than 30 minutes. For the real-life instances this tendency is even more apparent, since only 8 or 9 of the 80 instances were solved by CPLEX.

The exact algorithm was also unable to solve all of the instances, but performed significantly better than the CPLEX IP-solver as can be seen in Tables E.4 and E.5. For the packing instances, the exact algorithm solved between 51 and 58 of the 110 instances to optimality within the 30 minute time limit. For the 80 real-life instances, between 42 and 48 were solved. Moreover, for the instances where both algorithms solved the problem to optimality, the presented exact algorithm was considerably faster than CPLEX.

The complexity of the problem depends in large parts on the number of requests. The CPLEX IP-solver is able to solve all instances with 10 requests and most of the instances with 16 requests but none of the instances with more requests. The presented exact algorithm solves most of the instances with up to about 30 requests and a few with more than 30 requests.

It is interesting to notice that most of the considered instances are either solved very fast or not at all (given the imposed time limit). Considering that $\mathcal{L}_3$ was able to find the correct solution to almost every instance which was solved to optimality and in very little time, we expect that it is reasonably easy to find a good solution, but quite difficult to prove optimality. In many cases both the CPLEX IP-solver and the exact algorithm probably have found an optimal solution, but are unable to prove optimality within the given time frame.

Instance	Stations	N	Bin size 10 CPLEX	Bin size 10 B&B	Bin size 20 CPLEX	Bin size 20 B&B	Bin size 40 CPLEX	Bin size 40 B&B
CGCUT01	15	16	0.19(3)	**0(5)**	0.31(5)	**0.04(5)**	0.06(5)	**0(5)**
CGCUT02	40	23	-	**148.52(2)**	-	**0.29(5)**	-	**1.21(4)**
CGCUT03	40	62	**-**	**-**	**-**	**-**	**-**	**-**
GCUT01	250	10	0.4(5)	**0(5)**	0.13(5)	**0(5)**	0.48(5)	**0(5)**
GCUT02	250	20	-	**11.37(5)**	-	**1.01(5)**	-	**1.04(5)**
GCUT03	250	30	-	**781.71(2)**	-	**923.31(1)**	-	**482.18(1)**
GCUT04	250	50	**-**	**-**	**-**	**-**	**-**	**-**
GCUT05	500	10	2.34(5)	**0(5)**	1.36(5)	**0(5)**	1.97(5)	**0(5)**
GCUT06	500	20	-	**0.14(5)**	-	**0.33(5)**	-	**9.95(3)**
GCUT07	500	30	-	**8.08(2)**	-	**20.96(2)**	-	**1003.49(1)**
GCUT08	500	50	**-**	**-**	**-**	**-**	**-**	**-**
GCUT09	1000	10	1.63(5)	**0(5)**	1.15(5)	**0(5)**	3.06(5)	**0(5)**
GCUT10	1000	20	-	**0.32(5)**	-	**21.97(5)**	-	**58.28(5)**
GCUT11	1000	30	-	**644.54(2)**	**-**	**-**	-	**976.63(4)**
GCUT12	1000	50	**-**	**-**	**-**	**-**	-	**228.71(1)**
GCUT13	3000	32	-	**81.88(2)**	-	**75.02(4)**	-	**0.98(3)**
OKP01	100	50	**-**	**-**	**-**	**-**	**-**	**-**
OKP02	100	30	-	**33.15(2)**	-	**75.6(4)**	-	**60.03(2)**
OKP03	100	30	-	**310.31(4)**	**-**	**-**	-	**4.79(5)**
OKP04	100	61	-	**0.69(1)**	**-**	**-**	-	**0.4(2)**
OKP05	100	97	**-**	**-**	**-**	**-**	**-**	**-**
WANG20	70	42	**-**	**-**	**-**	**-**	-	**4.33(2)**
Average	484.77	35.14	1.14(18)	134.71(52)	0.74(20)	93.21(51)	1.39(20)	166.59(58)

Table E.4: Comparison between CPLEX IP-solver and the proposed branch-and-bound algorithm on the packing-inspired GSR-BPP instances. We report the instance name, the number of stations on the train route, the reservation count N, the average run time in seconds (and the number of instances solved within 30 minutes in parentheses) for each bin size. A dash indicates that no instances were solved. The best result for each bin size is shown in bold face.

Instance	Stations	N	Bin size 10 CPLEX	Bin size 10 B&B	Bin size 20 CPLEX	Bin size 20 B&B	Bin size 40 CPLEX	Bin size 40 B&B
G20N10_30	100	20	256.74(3)	**5.37(4)**	0.36(1)	**73.23(5)**	223.95(2)	**1.35(5)**
G20N20_20	100	20	-	**179.44(4)**	-	**119.81(5)**	-	**576.58(5)**
G20N30_10	100	20	-	**1.56(5)**	-	**11.12(5)**	-	**0.16(5)**
G20U20_20	100	20	485.27(5)	**33.76(5)**	232.59(5)	**0(5)**	14.24(5)	**0.55(5)**
G30N10_30	100	30	-	**3.26(5)**	-	**0.03(3)**	-	**4.06(3)**
G30N20_20	100	30	-	**263.17(3)**	-	**6.78(4)**	-	**252.99(1)**
G30N30_10	100	30	-	**2.16(2)**	-	**74.22(2)**	-	**110.77(3)**
G30U20_20	100	30	1721.43(1)	**25.03(5)**	45.29(1)	**0.09(5)**	1399.65(1)	**0.03(2)**
G40N10_30	100	40	-	**0.06(1)**	-	**0.06(2)**	-	**0.56(3)**
G40N20_20	100	40	-	**14.08(1)**	-	**740.75(2)**	-	**16.22(1)**
G40N30_10	100	40	**-**	**-**	-	**90.12(1)**	-	**0.63(2)**
G40U20_20	100	40	-	**116.75(3)**	365.68(1)	**2.54(2)**	-	**9.96(3)**
G50N10_30	100	50	-	**0.15(2)**	-	**0.13(2)**	-	**0.33(2)**
G50N20_20	100	50	-	**12.28(1)**	**-**	**-**	**-**	**-**
G50N30_10	100	50	-	**276.88(1)**	**-**	**-**	**-**	**-**
G50U20_20	100	50	-	**13.45(2)**	-	**5.18(5)**	-	**13.73(2)**
Average	100	35	821.15(9)	63.16(44)	160.98(8)	80.29(48)	545.95(8)	70.57(42)

Table E.5: Comparison between CPLEX IP-solver and the proposed branch-and-bound algorithm on the real-life-inspired GSR-BPP instances. We report the instance name, the number of stations on the train route, the reservation count N, the average run time in seconds (and the number of instances solved within 30 minutes in parentheses) for each bin size. A dash indicates that no instances were solved. The best result for each bin size is shown in bold face.

E.6 Further Work

The GSR-KP considers the profit of each reservation to be the product of the number of seats occupied and the distance traveled. A more general approach would be to represent the profit of a reservation as a separate parameter, as is the case in two-dimensional knapsack problems. In this way, route segments can be priced individually, which more realistically models the pricing presently done by many railway companies. However, the upper bounds presented cannot be used without modification.

For the GSR-BPP, no upper bounds have been considered. As the feasibility computations are much more efficient for the GSR-BPP than for ordinary two-dimensional packing problems, it is not known what effect an upper bound would have on the efficiency of the exact algorithm presented.

For the sub-problem of determining the feasibility of a packing, the implemented solution is based on graph structure properties and the solution of longest path problems. It is not known if improvements can be made by considering more recent bounds on packing feasibility problems in the literature, e.g. those of Clautiaux el al [13].

E.7 Conclusion

This is, to the best of our knowledge, the first paper to study the exact solution of the off-line GSRP. Two variants have been considered: the GSR-KP which is a special case of the 2DKP and the GSR-BPP which is a special case of the 2DBPP. For each problem, a number of bounds have been proposed and exact algorithms to solve the problems have been implemented. Additionally, necessary and sufficient conditions for feasible GSRP solutions based on a graph representation of the reservations have been described. An enumerative algorithm using these conditions has been implemented and is used in the exact algorithms of both the GSR-KP and the GSP-BPP.

For the GSR-KP we have proposed four upper bounds, which have been compared theoretically and computationally. Of these, the tightest bound and the fastest bound have been used in an exact algorithm. For comparison, the instances were also solved using the CPLEX IP-solver.

Of the implemented algorithms, the algorithm using the fastest bound showed the poorest performance, and was the fastest algorithm on very few instances.

The algorithm using the tightest bound solved all instances within reasonable time, and was significantly faster than the CPLEX IP-solver on some of the hardest instances. For many of the medium-sized instances, however, the CPLEX IP-solver showed the best performance.

For the GSR-BPP four lower bounds were considered. Of these, $\mathcal{L}_3$ was expected to perform well, and it was implemented with promising results. The GSR-BPP was more complex to solve than the GSR-KP. Both the CPLEX IP-solver and the exact algorithm implemented had difficulties solving some of the instances. Neither was able to solve all of the instances within a 30 minute time limit, but the exact algorithm solved several instances which the CPLEX IP-solver was unable to solve. For the instances solvable by the CPLEX IP-solver, the exact algorithm was the fastest in all cases. Thus for this version of the GSRP the presented algorithm is clearly the best choice, when an exact solution is desired.

Appendix A

Lemma 1

$a + b + c + d \leq 3$ when $ab + cd \leq 1$ and $0 \leq a, b, c, d \leq 1$

Proof. This is easily checked with a quadratic solver.

Proof of Theorem 2

We will show that every graph G that satisfies properties P1 and P2 corresponds to an arrangement of rectangles that comprises a feasible packing, i.e. the rectangles satisfies the following:

(1) None of the rectangles overlap

(2) All rectangles are placed within the box representing the train

To show (1) consider two nodes $i, j \in V$. Since G is acyclic (by P1), exactly one of the three following cases occur:

i) There is a path p from i to j in G.
Assume wlog. that p has length k and $p = \langle v_0, v_1, \ldots, v_k \rangle$ with $v_0 = i$ and

$v_k = j$. For each edge $(v_m, v_{m+1}) \in E$ we can place v_m and v_{m+1} so that $x_m + w_m \leq x_{m+1}$. By transitivity this will ensure $x_i + w_i \leq x_j$, yielding a packing where i and j does not overlap.

ii) There is a path from j to i in G.
This case is analogous to i) and yields $x_j + w_j \leq x_i$.

iii) There is no path between i and j in G.
i and j are positioned at different heights in the box, i.e. $y_i + h_i \leq y_j$ or $y_j + h_j \leq y_i$. Since the y-coordinates are fixed there is no way for i and j to overlap.

Since a packing without overlap exists for each of the three cases and i and j were chosen arbitrarily, (1) is proven.
To show (2) consider a rectangle $j \in N$ and assume for the sake of contradiction that $x_j + w_j > W$. Since $w_j \leq W$, $x_j > 0$. By assumption (E.22), j is positioned as far left as possible, so there must exist a rectangle $i \in N$ with $x_i + w_i = x_j$ and $(i, j) \in E$, i.e. rectangle i blocks j from being pushed further to the left. Similarly, another such rectangle will exist for i unless $x_i = 0$. This leads to a sequence of rectangles $\{m_0, m_1, \ldots, m_k\}$ where $j = m_0$ and $i = m_1$. Each rectangle in the sequence will touch its immediate successor and predecessor in the sequence.
By considering the sequence in reverse order (such that the rectangles will be ordered left to right), we get a path in G. Since the rectangles touch each other, $x_{m_s} = \sum_{t=s+1}^{k} w_t$. Rectangle j is the last rectangle on the path, so $x_j = m_0 = \sum_{t=1}^{k} w_t$. But $x_j + w_j = \sum_{v \in p} w_v \leq W$ by property P2, which leads to the desired contradiction. Since j was chosen arbitrarily, this proves (2). $\square$

References

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows*. Prentice Hall, New Jersey, 1993.

[2] Algorithmic Solutions Software GmbH, Germany. *The LEDA User Manual*, leda 4.5 edition.

[3] R. Baldacci and M. Boschetti. A cutting-plane approach for the two-dimensional orthogonal non-guillotine cutting problem. *European Journal of Operational Research*, 183:1136–1149, 2007.

[4] J. Beasley. Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society*, 36:297–306, 1985.

[5] G. Belov. *Problems, Models and Algorithms in One- and Two-Dimensional Cutting*. PhD thesis, Technischen Universität Dresden, 2003.

[6] J. Berkey and P. Wang. Two dimensional finite bin packing algorithms. *J. Oper. Res. Soc.*, 38:423–429, 1987.

[7] M. A. Boschetti and A. Mingozzi. Two-dimensional finite bin packing problem. part i: New lower bounds for the oriented case. *4OR: Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1:27–42, 2003.

[8] M. A. Boschetti and A. Mingozzi. Two-dimensional finite bin packing problem. part ii: New lower and upper bounds. *4OR: Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1:135–147, 2003.

[9] J. Boyar and P. Medvedev. The relative worst order ratio applied to seat reservation. In T. Hagerup and J. Katajainen, editors, *Proceedings of SWAT*

2004, volume 3111 of *Lecture Notes in Computer Science*, Heidelberg, 2004. Springer.

[10] A. Caprara and M. Monaci. On the two-dimensional knapsack problem. *Operations Research Letters*, 32:5–14, 2004.

[11] N. Christofides and C. Whitlock. An algorithm for two-dimensional cutting problems. *Operations Research*, 25:30–44, 1977.

[12] T. Clausen, A. Hjorth, M. Nielsen, and D. Pisinger. The off-line group seat reservation problem. Technical report, DIKU, University of Copenhagen, 2007.

[13] F. Clautiaux, C. Alves, and J. V. de Carvalho. A survey of dual feasible and superadditive functions. *Annals of Operations Research*, 2008. doi: 10.1007/s10479-008-0453-8.

[14] F. Clautiaux, A. Jouglet, J. Carlier, and A. Moukrim. A new constraint programming approach for the orthogonal packing problem. *Computers and Operations Research*, 35:944–959, 2008.

[15] J. F. Cordeau, M. Gaudioso, G. Laporte, P. Legato, and L. Moccia. Solving berth scheduling and yard management problems at the gioia tauro maritime terminal, 2003.

[16] J. Csirik and G. Woeginger. On-line packing and covering problems. In A. Fiat and G. Woeginger, editors, *Online algorithms*, volume 1442 of *Lecture Notes in Computer Science*, pages 147–177. Springer, Heidelberg, 1998.

[17] M. Dell'Amico and S. Martello. Optimal scheduling of tasks on identical parallel processors. *ORSA Journal on Computing*, 7:191–200, 1995.

[18] S. Fekete and J. Schepers. New classes of fast lower bounds for bin packing problems. *Mathematical Programming*, 91:11–31, 2001.

[19] S. Fekete and J. Schepers. A combinatorial characterization of higher-dimensional packing. *Mathematics of Operations Research*, 29:353–368, 2004.

[20] S. Fekete, J. Schepers, and J. van der Veen. An exact algorithm for higher-dimensional orthogonal packing. *Operations Research*, 55:569–587, 2007.

[21] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.

[22] E. Hadjiconstantinou and N. Christofides. An exact algorithm for general, orthogonal, two-dimensional knapsack problems. *European Journal of Operational Research*, 83:39–56, 1995.

[23] A. Helliesen. The seat reservation problem - an empirical study. Master's thesis, Technical University of Denmark, October 2003.

[24] A. Imai, J.-T. Zhang, E. Nishimura, and S. Papadimitriou. The berth allocation problem with service time and delay time objectives. *Maritime Economics & Logistics*, 9:269–290, 2007.

[25] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, Heidelberg, 2004.

[26] A. Lodi, S. Martello, and D. Vigo. Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics*, 123:379–396, 2002.

[27] S. Martello and D.Vigo. Exact solution of the two-dimensional finite bin packing problem. *Manage. Sci.*, 44:388–399, 1998.

[28] S. Martello, D. Pisinger, and D. Vigo. The three-dimensional bin packing problem. *Operations Research*, 48:256–267, 2000.

[29] S. Martello and P. Toth. Lower bounds and reduction procedures for the bin packing problem. *Discrete Applied Mathematics*, 28:59–70, 1990.

[30] M. Pal and G. P. Bhattacharjee. A sequential algorithm for finding a maximum weight k-independent set on interval graphs. *Intern. J. Computer Math.*, 60:205–214, 1996.

[31] P. Pandey, P. Yenradee, and S. Archariyapruek. A finite capacity material requirements planning system,. *Production Planning and Control*, 11:113–121, 2000.

[32] D. Pisinger and M. Sigurd. Using decomposition techniques and constraint programming for solving the two-dimensional bin packing problem. *INFORMS Journal on Computing*, 19:36–51, 2007.

[33] P. Y. Wang. Two algorithms for constrained two-dimensional cutting stock problems. *Operations Research*, 31:573–586, 1983.

[34] M. Yannakakis and F. Gavril. The maximum k-colorable subgraph problem for chordal graphs. *Information Processing Letters*, 24:133–137, 1987.

Bibliography

[1] A. Abdelghany, K. Abdelghany, and R. Narasimhan. Scheduling baggage-handling facilities in congested airports. *Journal of Air Transport Management*, 12(2):76–81, 2006.

[2] I. Addou and F. Soumis. Bechtold-Jacobs generalized model for shift scheduling with extraordinary overlap. *Annals of Operations Research*, 155(1):177–205, 2007.

[3] H. Alfares. Aircraft maintenance workforce scheduling: a case study. *Journal of Quality in Maintenance Engineering*, 5(2):78–88, 1999.

[4] H. Alfares. Survey, categorization, and comparison of recent tour scheduling literature. *Annals of Operations Research*, 127(1):145–175, 2004.

[5] H. Alfares and J. Bailey. Integrated project task and manpower scheduling. *IIE transactions*, 29(9):711–717, 1997.

[6] N. Ashford, H. Stanton, and C. Moore. *Airport operations*. McGraw-Hill Professional, 1998.

[7] J. Atkin, E. Burke, J. Greenwood, and D. Reeson. A Metaheuristic Approach to Aircraft Departure Scheduling at London Heathrow Airport. *Computer-aided Systems in Public Transport*, pages 235–252, 2008.

[8] C. Azmat and M. Widmer. A case study of single shift planning and scheduling under annualized hours: A simple three-step approach. *European Journal of Operational Research*, 153(1):148–175, 2004.

[9] BAA. Economic benefits of aviation. Issue Brief, 2004.

[10] O. Babić and D. Teodorović. Aircraft stand assignment to minimize walking. *Journal of Transportation Engineering*, 110:55, 1984.

[11] J. Bailey. Integrated days off and shift personnel scheduling. *Computers & Industrial Engineering*, 9(4):395–404, 1985.

[12] K. Baker. Workforce allocation in cyclical scheduling problems: A survey. *Operational Research Quarterly*, 27(1):155–167, 1976.

[13] J. Beasley, J. Sonander, and P. Havelock. Scheduling aircraft landings at London Heathrow using a population heuristic. *Journal of the Operational Research Society*, 52(5):483–493, 2001.

[14] S. Bechtold and M. Brusco. A microcomputer-based heuristic for tour scheduling of a mixed workforce. *Computers & Operations Research*, 21(9):1001–1009, 1994.

[15] A. Beer, J. Gärtner, N. Musliu, W. Schafhauser, and W. Slany. Scheduling Breaks in Shift Plans for Call Centers. In *The 7th International Conference on the Practice and Theory of Automated Timetabling*, 2008.

[16] A. Billionnet. Integer programming to schedule a hierarchical workforce with variable demands. *European Journal of Operational Research*, 114(1):105–114, 1999.

[17] Y. Borenstein, N. Shah, E. Tsang, R. Dorne, A. Alsheddy, and C. Voudouris. On the partitioning of dynamic workforce scheduling problems. *Journal of Scheduling*, pages 1–15, 2009.

[18] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1):3–41, 1999.

[19] M. Brusco and L. Jacobs. A simulated annealing approach to the solution of flexible labour scheduling problems. *Journal of the Operational Research Society*, 44(12):1191–1200, 1993.

[20] M. Brusco and L. Jacobs. Personnel tour scheduling when starting-time restrictions are present. *Management Science*, 44(4):534–547, 1998.

[21] M. Brusco and L. Jacobs. Optimal models for meal-break and start-time flexibility in continuous tour scheduling. *Management Science*, pages 1630–1641, 2000.

[22] M. Brusco and L. Jacobs. Starting-time decisions in labor tour scheduling: An experimental analysis and case study. *European Journal of Operational Research*, 131(3):459–475, 2001.

[23] M. Brusco, L. Jacobs, R. Bongiorno, D. Lyons, and B. Tang. Improving personnel scheduling at airline stations. *Operations Research*, 43(5):741–751, 1995.

[24] M. Brusco and T. Johns. Improving the dispersion of surplus labor in personnel scheduling solutions. *Computers & Industrial Engineering*, 28(4):745–754, 1995.

[25] E. Burke, P. De Causmaecker, G. Berghe, and H. Van Landeghem. The state of the art of nurse rostering. *Journal of scheduling*, 7(6):441–499, 2004.

[26] A. Caprara, M. Fischetti, P. Toth, D. Vigo, and P. Guida. Algorithms for railway crew management. *Mathematical Programming*, 79(1):125–141, 1997.

[27] A. R. Center. Study on the impact of directive 96/67/ec on ground handling services 1996-2007, 2009.

[28] I. Chao, B. Golden, and E. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88(3), 1996.

[29] Z. Chen and H. Xu. Dynamic column generation for dynamic vehicle routing with time windows. *Transportation Science*, 40(1):74–88, 2006.

[30] Y. Cheng. Solving push-out conflicts in apron taxiways of airports by a network-based simulation. *Computers & Industrial Engineering*, 34(2):351–369, 1998.

[31] K. Chew. Cyclic schedule for apron services. *Journal of the Operational Research Society*, 42(12):1061–1069, 1991.

[32] S. Chu. Generating, scheduling and rostering of shift crew-duties: Applications at the Hong Kong International Airport. *European Journal of Operational Research*, 177(3):1764–1778, 2007.

[33] S. Chu and C. Lin. A Manpower Allocation Model of Job Specialization. *Journal of the Operational Research Society*, 44(10):98–989, 1993.

[34] H. Chun. Scheduling as a multi-dimensional placement problem. *Engineering Applications of Artificial Intelligence*, 9(3):261–273, 1996.

[35] J. Clausen, A. Larsen, J. Larsen, and N. Rezanova. Disruption management in the airline industry–Concepts, models and methods. *Computers & Operations Research*, 37(5):809–821, 2010.

[36] J. Cordeau and G. Laporte. The dial-a-ride problem (DARP): Variants, modeling issues and algorithms. *4OR: A Quarterly Journal of Operations Research*, 1(2):89–101, 2003.

[37] M. Côté, B. Gendron, C. Quimper, and L. Rousseau. Formal languages for integer programming modeling of shift scheduling problems. *Constraints*, pages 1–23, 2007.

[38] P. Cowling, N. Colledge, K. Dahal, and S. Remde. The trade off between diversity and quality for multi-objective workforce scheduling. *Evolutionary Computation in Combinatorial Optimization*, pages 13–24, 2006.

[39] G. B. Dantzig. A comment on edie's "traffic delays at toll booths". *Journal of the Operations Research Society of America*, 2(3):339–341, 1954.

[40] L. Di Gaspero, J. Gärtner, G. Kortsarz, N. Musliu, A. Schaerf, and W. Slany. The minimum shift design problem. *Annals of Operations Research*, 155(1):79–105, 2007.

[41] L. Di Gaspero, J. Gärtner, N. Musliu, A. Schaerf, W. Schafhauser, and W. Slany. A Hybrid LS-CP Solver for the Shifts and Breaks Design Problem. In *The 7th International Workshop on Hybrid Metaheuristics (HM 2010). Lecture Notes in Computer Science. To appear, Vienna, Austria*, 2010.

[42] G. Diepen, J. M. V. D. Akker, and J. A. Hoogeveen. Integrated gate and bus assignment at amsterdam airport schiphol. Technical Report UU-CS-2008-041, Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands, 2008.

[43] G. Diepen, J. v. d. Akker, and J. H. e. J. Smeltink. Using column generation for gate planning at amsterdam airport schiphol. Technical Report UU-CS-2007-018, Department of Information and Computing Sciences, Utrecht University, 2007.

[44] A. Dohn, E. Kolind, and J. Clausen. The manpower allocation problem with time windows and job-teaming constraints: A branch-and-price approach. *Computers & Operations Research*, 36(4):1145–1157, 2009.

[45] U. Dorndorf. Staff and Resource Scheduling at Airports. *Operations Research Proceedings 2006*, pages 3–7, 2007.

[46] D. Dowling, M. Krishnamoorthy, H. Mackenzie, and D. Sier. Staff rostering at a large international airport. *Annals of Operations Research*, 72(0):125–147, 1997.

[47] C. Duin and E. Der Sluis. On the complexity of adjacent resource scheduling. *Journal of Scheduling*, 9(1):49–62, 2006.

[48] A. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European journal of operational research*, 153(1):3–27, 2004.

[49] European Commission. The eu ensures access to air transport for persons with reduced mobility. Press Release IP/07/1173, 2007.

[50] G. Felici and C. Gentile. A polyhedral approach for the staff rostering problem. *Management Science*, 50(3):381–393, 2004.

[51] B. Fleischmann, S. Gnutzmann, and E. Sandvoß. Dynamic vehicle routing based on online traffic information. *Transportation science*, 38(4):420–433, 2004.

[52] G. Ghiani, F. Guerriero, G. Laporte, and R. Musmanno. Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *European Journal of Operational Research*, 151(1):1–11, 2003.

[53] A. Goel, V. Gruhn, and T. Richter. Mobile Workforce Scheduling Problem with Multitask-Processes. In *Business Process Management Workshops*, pages 81–91. Springer, 2010.

[54] B. Golden, S. Raghavan, and E. Wasil. *The vehicle routing problem: latest advances and new challenges*. Springer, 2008.

[55] J. Goto, M. Lewis, and M. Puterman. Coffee, tea, or...?: A markov decision process model for airline meal provisioning. *Transportation Science*, 38(1):107–118, 2004.

[56] A. Haghani and M. Chen. Optimizing gate assignments at airport terminals. *Transportation Research Part A: Policy and Practice*, 32(6):437–454, 1998.

[57] J. Herbers. *Models and Algorithms for Ground Staff Scheduling On Airports*. Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, Faculty of Mathematics, Computer Science and Natural Sciences, 2005.

[58] J. Herbers. Representing Labor Demands in Airport Ground Staff Scheduling. *Operations Research Proceedings 2005*, pages 15–20, 2006.

[59] S. Ho and J. Leung. Solving a manpower scheduling problem for airline catering using metaheuristics. *European Journal of Operational Research*, 202(3):903–921, 2010.

[60] K. Hoffman and M. Padberg. Solving airline crew scheduling problems by branch-and-cut. *Management Science*, 39(6):657–682, 1993.

[61] R. Hung. Single-shift off-day scheduling of a hierarchical workforce with variable demands. *European Journal of Operational Research*, 78(1):49–57, 1994.

[62] IATA. Financial forecast, Jun 2010.

[63] O. Jabali, T. Van Woensel, A. de Kok, C. Lecluyse, and H. Peremans. Time-dependent vehicle routing subject to time delay perturbations. *IIE Transactions*, 41(12):1049–1066, 2009.

[64] L. Jacobs and M. Brusco. Overlapping start-time bands in implicit tour scheduling. *Management Science*, pages 1247–1259, 1996.

[65] E. Keith. Operator scheduling. *IIE Transactions*, 11(1):37–41, 1979.

[66] N. Kohl and S. Karisch. Airline crew rostering: problem types, modeling, and optimization. *Annals of Operations Research*, 127(1):223–257, 2004.

[67] N. Kohl, A. Larsen, J. Larsen, A. Ross, and S. Tiourine. Airline disruption management–Perspectives, experiences and outlook. *Journal of Air Transport Management*, 13(3):149–162, 2007.

[68] R. Kolisch and S. Hartmann. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 174(1):23–37, 2006.

[69] A. Larsen, O. Madsen, and M. Solomon. The a priori dynamic traveling salesman problem with time windows. *Transportation Science*, 38(4):459–472, 2004.

[70] A. Larsen, O. Madsen, and M. Solomon. Recent developments in dynamic vehicle routing systems. *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 199–218, 2008.

[71] H. C. Lau. On the complexity of manpower shift scheduling. *Computers & Operations Research*, 23(1):93–102, 1996.

[72] Y. Li, A. Lim, and B. Rodrigues. Manpower allocation with time windows and job-teaming constraints. *Naval Research Logistics*, 52(4):302–311, 2005.

[73] A. Lim, B. Rodrigues, and L. Song. Manpower allocation with time windows. *Journal of the Operational Research Society*, 55(11):1178–1186, 2004.

[74] C. K. Y. Lin, K. F. Lai, and S. L. Hung. Development of a workforce management system for a customer hotline service. *Computers & Operations Research*, 27(10):987–1004, 2000.

[75] J. Loucks and F. Jacobs. Tour scheduling and task assignment of a heterogeneous work force: a heuristic approach. *Decision Sciences*, 22(4):719–738, 1991.

[76] R. Lusby, A. Hansen, T. Range, and J. Larsen. An Integrated Approach to the Ground Crew Rostering Problem with Work Patterns. Technical report, DTU Management Kgs. Lyngby, 2010.

[77] A. Mason, D. Ryan, and D. Panton. Integrated simulation, heuristic and optimisation approaches to staff scheduling. *Operations Research*, pages 161–175, 1998.

[78] L. McGinnis, W. Culver, and R. Deane. One-and two-phase heuristics for workforce scheduling. *Computers & Industrial Engineering*, 2(1):7–15, 1978.

[79] M. Mederer, G. Klempert, and T. Arzt. De-peaking Lufthansa Hub Operations at Frankfurt Airport. In M. Rabe, editor, *Advances in Simulation for Production and Logistics Applications*. Fraunhofer IRB Verlag, 2008.

[80] N. Musliu, J. Gärtner, and W. Slany. Efficient generation of rotating workforce schedules. *Discrete Applied Mathematics*, 118(1-2):85–98, 2002.

[81] N. Musliu, A. Schaerf, and W. Slany. Local search for shift design. *European Journal of Operational Research*, 153(1):51–64, 2004.

[82] R. Nissen and K. Haase. Duty-period-based network model for crew rescheduling in European airlines. *Journal of Scheduling*, 9(3):255–278, 2006.

[83] W. Powell, M. Towns, and A. Marar. On the value of optimal myopic solutions for dynamic routing and scheduling problems in the presence of user noncompliance. *Transportation Science*, 34(1):67–85, 2000.

[84] H. Psaraftis. Dynamic vehicle routing: Status and prospects. *Annals of Operations Research*, 61(1):143–164, 1995.

[85] C. Quimper and L. Rousseau. A large neighbourhood search approach to the multi-activity shift scheduling problem. *Journal of Heuristics*, 16(3):373–392, 2010.

[86] A. Regan, H. Mahmassani, and P. Jaillet. Improving efficiency of commercial vehicle operations using real-time information: potential uses and assignment strategies. *Transportation Research Record*, 1493:188–198, 1995.

[87] M. Rekik, J. Cordeau, and F. Soumis. Implicit shift scheduling with multiple breaks and work stretch duration restrictions. *Journal of Scheduling*, 13(1):49–75, 2010.

[88] S. Schindler and T. Semmel. Station staffing at pan american world airways. *Interfaces*, 23(3):91–98, 1993.

[89] J. Schönberger. Adaptive demand peak management in online transport process planning. *OR Spectrum*, 32(3):831–859, 2010.

[90] M. Sørensen and J. Clausen. Decentralized ground staff scheduling. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, Kongens Lyngby, Denmark, 2002.

[91] H. Stern and M. Hersh. Scheduling aircraft cleaning crews. *Transportation Science*, 14(3):277, 1980.

[92] R. Stolletz. Operational workforce planning for check-in counters at airports. *Transportation Research Part E: Logistics and Transportation Review*, 2009.

[93] G. Thompson. Shift scheduling in services when employees have limited availability: an LP approach. *Journal of Operations Management*, 9(3):352–370, 1990.

[94] G. Thompson. Labor scheduling using NPV estimates of the marginal benefit of additional labor capacity. *Journal of Operations Management*, 13(1):67–86, 1995.

[95] G. Thompson. A simulated-annealing heuristic for shift scheduling using non-continuously available employees. *Computers & Operations Research*, 23(3):275–288, 1996.

[96] G. Thompson and J. Goodale. Variable employee productivity in workforce scheduling. *European Journal of Operational Research*, 170(2):376–390, 2006.

[97] J. Tien and A. Kamiyama. On manpower scheduling algorithms. *Siam Review*, 24(3):275–287, 1982.

[98] P. Toth and D. Vigo. *The vehicle routing problem*. Society for Industrial Mathematics, 2002.

[99] UK Civil Aviation Authority. Summary of activity at uk airports 2009, 2010.

[100] P. Vance, C. Barnhart, E. Johnson, and G. Nemhauser. Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research*, 45(2):188–200, 1997.

[101] G. Vanderstraeten and M. Bergeron. Automatic assignment of aircraft to gates at a terminal. *Computers & industrial engineering*, 14(1):15–25, 1988.

[102] M. Wen, J. Larsen, and J. Clausen. An exact algorithm for aircraft landing problem. Technical Report IMM-Technical Report-2005-12, Informatics and Mathematical Modelling, Technical University of Denmark, 2005.

[103] M. Widl and N. Musliu. An Improved Memetic Algorithm for Break Scheduling. In *Proceedings of the 7th International Workshop on Hybrid Metaheuristics (HM 2010). Lecture Notes in Computer Science*, 2010.

[104] A. Wren and J. Rousseau. Bus driver scheduling-an overview. In J. R. Daduna, I. Branco, and J. M. P. Paixao, editors, *Computer-Aided Transit Scheduling: Proceedings of the Sixth International Workshop on Computer-Aided Scheduling of Public Transport*, volume 430 of *Lecture Notes in Economics and Mathematical Systems*, pages 173–187, 1993.

[105] G. Yu, M. Arguello, G. Song, S. McCowan, and A. White. A new era for crew recovery at Continental Airlines. *Interfaces*, 33(1):5, 2003.

[106] G. Yu and X. Qi. *Disruption management: framework, models and applications*. World Scientific Pub Co Inc, 2004.

Modern airports are centers of transportation that service a large number of aircraft and passengers every day. When an aircraft lands, a significant number of tasks must be performed by different groups of ground crew before the aircraft departs, such as fueling, baggage handling and cleaning. These tasks are collectively known as ground handling and are the major source of activity with airports.

The thesis contains an introductory part which provide an overview of the ground handling environment and reviews a series of optimization problems from the specific perspective of airport ground handling. Problems considered range from generalized approaches to workforce planning, to detailed scheduling problems arising in the highly dynamic environment of airports.

ISBN 978-87-90855-95-6

DTU Management Engineering
Department of Management Engineering
Technical University of Denmark

Produktionstorvet
Building 424
DK-2800 Kongens Lyngby
Denmark
Tel. +45 45 25 48 00
Fax +45 45 93 34 35

www.man.dtu.dk