A Population Based Variable Neighborhood Search algorithm for the Minimum Shift Design problem

Master thesis

Economics and Informatics
Erasmus School of Economics
Erasmus University Rotterdam

1st Supervisor: U. Kaymak
2nd Supervisor: E. van Asperen

C.M. Ng 267625

March 17, 2010

Preface

This thesis is the final step of my study Economics and Informatics at the Erasmus University Rotterdam. I've written this thesis for my Master program Computational Economics. In this thesis, I focus on creating an algorithm for designing shifts at a tactical level for the company TNT Express.

A couple of years ago, I applied for a student position at ORTEC. It was also during this time that I wrote my Bachelor thesis at that same company. Last year, I did the Master program while still working two days a week at ORTEC. Eventually, it was time for me to write my master thesis and I decided to also write it at ORTEC. It has been a long journey and I would like to thank all my colleagues for all the support. Some of the people I would like to thank personally. First of all Arjen van de Wetering for giving me the opportunity to work and write my thesis at ORTEC. Secondly Frank van der Wal, for helping me and giving me guidance during the writing of this thesis. Finally, all my other colleagues from the development team at ORTEC: Patrick Hennen, Daan Noorlander, Jarco Visscher and Dragos Tihauan.

At last, I would like to thank some other people for their contribution and support. I would like to thank my supervisors from the university, Uzay Kaymak and Eelco van Asperen. Your knowledge and feedback have been of great added value for my thesis. I would also like to thank my family for their support and patience. Last but not least, I would like to thank Floortje and Hubert for always supporting me and believing in me.

C.M. Ng
Rotterdam, March 2010

Abstract

For many companies in different industries to be able to reduce their costs and improve on their efficiency, it is necessary for them to make use of optimized shift schedules and rosters. In this thesis we propose to use the *Min*-Shift Design (MSD) problem formulation to create workforce schedules at a tactical level for an express company called TNT Express. At this company, the estimated workforce requirements for the coming planning horizon are known beforehand. The goal is to be able to create a tactical workforce schedule based on the estimated workforce requirements that covers all estimated workload. This tactical workforce schedule should specify the start and end times of each shift to be used and also the number of employees needed for each such shift.

To solve the MSD problem, we developed a new solution method called Population Based Variable Neighborhood Search (PBVNS). This method is based on the principles of Variable Neighborhood Search and Evolutionary Algorithms. By combining some of the key elements of both methods, we developed an algorithm that can be used to solve synthetic and real world instances of shift design problems.

Keywords: Shift design, Shift scheduling, Variable Neighborhood Search, Evolutionary Algorithm.

Contents

1 Introduction **3**

2 Shift scheduling: applications and solution methods **8**
- 2.1 Introduction . . . 8
- 2.2 Application areas . . . 9
  - 2.2.1 Transportation systems . . . 9
  - 2.2.2 Call centers . . . 10
  - 2.2.3 Health care systems . . . 11
  - 2.2.4 Protection and emergency services . . . 11
  - 2.2.5 Other applications . . . 12
- 2.3 Solution methods . . . 12
  - 2.3.1 Mathematical Programming . . . 12
  - 2.3.2 Meta-heuristics . . . 13
- 2.4 Recommendations for designing algorithms . . . 16

3 Minimum Shift Design Problem **18**
- 3.1 Background . . . 18
- 3.2 Problem definition . . . 18
- 3.3 Solving the mathematical model . . . 21

4 Variable Neighborhood Search **23**
- 4.1 Introduction . . . 23
- 4.2 Variable Neighborhood Search . . . 25
  - 4.2.1 Reduced Variable Neighborhood Search . . . 26
  - 4.2.2 General Variable Neighborhood Search . . . 27
- 4.3 Fitness function . . . 28
- 4.4 Neighborhood structures . . . 28
- 4.5 Initial solution . . . 29
- 4.6 Processing steps . . . 29
  - 4.6.1 Delete shifts . . . 30
  - 4.6.2 Insert shifts . . . 30
- 4.7 Overview of proposed VNS algorithm . . . 30
- 4.8 Population Based VNS . . . 31
  - 4.8.1 Motivation . . . 31

4.8.2 Modifications 32
4.8.3 Outline of PBVNS 33

5 Experiments and computational results 35
5.1 Experimental Setup 35
5.1.1 Performance measures 35
5.1.2 ROTA test examples 36
5.1.3 Experiments and parameters 37
5.2 VNS vs PBVNS 39
5.3 Computational results 40
5.3.1 Population size 40
5.3.2 Reduced Variable Neighborhood Search 41
5.3.3 Number of initial shifts and employees 41
5.3.4 Shake phase in VNS 41
5.3.5 Best results 44
5.3.6 Comparison with another method from the literature . . . 45
5.4 Real world case 46
5.4.1 Experimental setup 48
5.4.2 Computational results 49
5.4.3 Best results real world case 50
5.5 Managerial implications 52

6 Conclusion and future research 53

Chapter 1

Introduction

Workforce scheduling problems can be found in many different companies and industries. Whether it's manufacturing, call centers, airline industry, hospitals, distribution centers or transportation industry, it is always necessary to determine the number of workers that are needed in order to meet the total demand or a certain service level. The usage of optimized shifts and rosters, makes it possible to use the available resources as best as possible. This efficiency results in more satisfied customers and cost reduction.

For most companies, the process of creating a workforce planning or schedule is a difficult task. Often this process is divided into several stages. In the first stage, an estimation is made for the total workload that is required for each time slot in the planning horizon. This estimation can be based on forecasting techniques or expert knowledge of managers. In certain industries the workload is fully known. As in the airline industry, the timetables are known in advance and the number of crew members required can thus be fully determined. Another possibility is that by using an automated system, a company can determine the time of arrival of certain load to a specified location. The estimated workload is referred to in the literature as the temporal staff requirements. An example of this is: on Tuesday between 12:00 and 16:00, there are six employees necessary to complete all the work. After the first stage of creating a workforce planning, one can go to the next stages. That is to design the shifts and determining the number of employees that are needed for each shift. Finally, the designed shifts are assigned to the employees and thus a complete schedule is created.

ORTEC & TNT

This thesis is a result of an internship at ORTEC. ORTEC is a consulting company which focuses on applications of IT and operations research on financial and logistic markets. One of the key accounts of ORTEC is TNT. TNT is an express service provider and offers a wide range of mail and express distribution services worldwide for businesses and consumers. TNT consists of the two divisions TNT Post and TNT Express. The Express division transports 4,4 million packages, documents and freight to over two hundred countries on the weekly basis. TNT Express has a huge network; roughly 2,300 depots, airports, sorting centers and a fleet of almost 27,000 vehicles and 35 airplanes. With this, TNT Express possesses the largest country and air infrastructure for delivery services in Europe. ORTEC assists TNT in all kinds of planning problems they encounter with consulting services and tailor made software solutions. One of these software solutions is called the Manpower Planning TNT (MPT). This software is used to evaluate and optimize workforce schedules.

Manpower planning at TNT

Nowadays, a lot of scheduling and manpower planning software packages are available to help companies with their business. At TNT Express, a manpower planning system (MPT) is used to create a workforce schedule at a tactical level. The reason for TNT Express to use this system was twofold. First, they wanted to reduce the number of redundant employees. Secondly, they wanted to determine the best start and end times for the different shifts they plan. This system doesn't take the skill level of the employees into account, thus each employee is treated identically. The way their manpower planning system works is as follows. The temporal workforce requirements and the set of shifts for the planning horizon are entered into the system as input. Afterwards, the system automatically determines which shifts need to be used and also the optimal number of employees that need to be assigned to each of these shifts. Finally, the system returns the optimized schedule and shows the difference in man hours that have been reduced by optimizing the current schedule.

One of the main drawbacks of this system, is that it takes the set of shifts as an input. This means that the user of the system is actually only optimizing the schedule using the shift times he has entered. The start and end times of the shifts can not be modified during the optimization phase. The probability that the user has entered the set of shifts with the right times to create the most optimal schedule is quite low. As a work around for this issue, a lot of users of this system are entering a lot of shifts into the system. Some of these shifts only differ slightly from each other. For example, instead of entering one shift from 9:00 to 17:00, it can be the case that the shifts: 8:00 to 16:00, 8:00 to 17:00, 9:00 to 16:00, 9:00 to 17:00, etc. are all entered. The user hopes that the system will be able to identify those shifts that are needed to create the optimal schedule. But by entering a lot more shifts into the system, makes the problem itself much harder to solve. The reason for this is that the number of deterministic variables have also increased substantially. This results in much higher computation times and problems that can not be solved by the manpower planning system. Therefore, it is necessary that a new algorithm is developed for the manpower planning system that can create an optimized schedule without taking the set of shifts as input.

Designing shifts

In the literature there are a lot of publications concerning shift scheduling. Most of the times, the problem at hand is to determine the optimal number of employees to be assigned to predefined shifts in order to satisfy the demand. [16] presents a very detailed overview of the literature on staff scheduling and rostering problems. There are two main approaches to create a complete workforce schedule [21]. The first of these approaches tries to solve the design of the shifts and the assignment of the shifts to the employees as one single problem. The other approach is to design the shifts first and to schedule the actual employees afterwards. The main drawback of the latter approach is that there is no guarantee that a good assignment can be found after the shifts have been designed. However, dividing the problem into sub-problems makes the problem easier to solve. In this thesis we choose the latter approach. The primary focus is on the design of shifts and determining the number of employees necessary for each shift. The assignment of the actual employees to the different shifts will not be considered in this thesis.

[21] was the first to formally introduce the shift design problem. The shift design problem is the problem of finding a minimum set of work shifts to use and the number of workers to assign to each shift, in order to meet pre-specified staff requirements. In [21], the shift design problem was formulated as a multi-objective optimization problem where three components with different weights needed to be minimized. The first two components are the sum of the excesses and shortages of workers in each time slot during the planning period and the third component is the total number of shifts.

The proposed shift design problem formulation was solved using a local search that is based on the principles of tabu search. Furthermore, they proposed a procedure to generate a good initial solution and also a method that exploits domain knowledge during the search. The computational results show that their algorithm works well, but the authors admit that the local search can be further refined.

In [14], a similar problem was tackled. Here the problem was called the *min*-SHIFT DESIGN problem (MSD). The objective of this problem is the same as for the shift design problem formulated in [21]. However, [14] established the complexity of MSD by means of a reduction to a Network Flow problem, namely the cyclic multi-commodity capacitated fixed-charge *min*-COST *max*-FLOW problem. By neglecting the issue of minimizing the number of shifts, the MSD problem becomes solvable in polynomial time. The authors proposed a hybrid heuristic algorithm composed of a constructive heuristic and a multi-neighborhood tabu search procedure. The computational results show that their algorithm produces good solutions both in terms of ability to reach good solutions and in terms of computation time. They also show that their hybrid algorithm outperforms the local search proposed by [21].

[21] and [14] are two of the few publications in the literature about the shift design problem. This means that unlike other well-known problems in the literature such as the Vehicle Routing Problem (VRP), Traveling Salesman Problem

(TSP), Facility Location Problem (FLP) and many others, a lot of areas in this problem still need to be explored. Because of this, it is a very interesting topic for research purposes. The main contribution of this thesis is that we developed a new algorithm to solve the shift design problem. This algorithm is based on the Variable Neighborhood Search (VNS) framework. There have been many applications of VNS on scheduling problems in the literature. However, VNS has not been applied on shift design problems before. Also, the basic VNS scheme was not good enough to generate good quality solutions, thefore we extended the basic VNS framework by creating a population based VNS method.

Motivation

The motivation for this study is the increasing popularity of using the so called (meta)-heuristics to solve complex optimization problems. In the last decades, methods such as Simulated Annealing (SA), TABU search, Genetic Algorithm (GA) and Ant Colony Optimization (ACO) have been studied extensively to solve complex optimization problems. These heuristics have proven to have the ability to generate near optimal solutions. Apart from being robust, the computation times of these heuristics are also often quite short compared to exact methods. For an overview of heuristic methods and their applications we advise the reader to refer to [12] and [6]. As stated in [13], the shift design problem is NP-hard. Exact methods can be used to solve instances of this problem. However, there are some drawbacks when using exact methods. The first drawback is that exact methods are often problem specific. Secondly, using an exact method on large sized problems is computationally very expensive and the system can sometimes even run out of memory. Another drawback of using exact methods is that sometimes they require a lot of parameter settings. This makes it very difficult to develop a method with the right settings that is robust to a lot of different instances of the shift design problem. To overcome the issues from the exact methods, we have chosen for a heuristic approach. The challenge is to develop a heuristic that is robust to different instances of the shift design problem while having as few parameters as possible. Since VNS has shown to be a good solution method for some scheduling problems and it has very few parameters, our choice has fallen on this heuristic.

Objective and research question

The objective of this research is to design a search algorithm for solving (real world) cases of the minimum shift design problem. The first requisite for this algorithm is that it needs to be fast. The reason for this is that shift scheduling is a decision on the tactical level. Therefore a manager must be able to use this algorithm on a regular basis and analyze different scenarios to make sure that he has the right workforce schedule for the next planning horizon. Secondly, this algorithm must have as few parameters as possible. Since most people working with a shift scheduling system won't have the time and expertise to be playing with the parameters in order to get a good feasible schedule, they will want an algorithm that is easy to use and that gives good results. Finally, the proposed algorithm must support both next day planning as well as next week planning. Shift design problems for next day planning can sometimes already be quite hard. When looking over a week, the problem becomes even more complex. But there are a lot of businesses that go on for 24 hours a day and 7 days a week. We also want to be able to solve the shift design problem for these companies. However, if an algorithm is able to solve a problem for a week, then it should also be able to solve an one day problem. Based on the objectives that were just mentioned, the following research question is formulated

- How to design a search algorithm for the minimum shift design problem?

The following sub-questions are derived from the main research question:

- How to make the proposed algorithm efficient in terms of computation time?
- How to design an algorithm with the least parameters possible?
- How to deal with the constraints of the shift design problem?

Methodology

The methodology that was used in this thesis was as follows

- Firstly, we studied the scientific literature on shift design, workforce planning and shift scheduling problems.
- Secondly, we devised a mathematical formulation for the minimum shift design problem using models in the literature.
- Next, we developed an algorithm to solve this mathematical formulation.
- Then, the proposed algorithm was tested on examples of minimum shift design problems that were used in [21] and [14].
- Finally, the proposed algorithm was also used to solve a real world case of the shift design problem.

Outline thesis

The rest of this thesis is structured as follows. In Chapter 2, a literature review on shift scheduling problems is presented. Chapter 3 gives a formal description and mathematical formulation of the minimum shift design problem. A complete description of the algorithm used to solve the problem is outlined in Chapter 4. The computational results for the test examples are presented in Chapter 5. A comparison on the performance of the proposed algorithm with another algorithm in the literature will be presented here. Our proposed algorithm is also used to solve a real world instance of the shift design problem. Finally, Chapter 6 draws the conclusions and proposes some future research.

Chapter 2

Shift scheduling: applications and solution methods

2.1 Introduction

Staff scheduling and rostering decisions are often taken for a certain planning horizon. It is common that the process of staff scheduling can be divided into three main phases: staffing, shift/roster design and shift/roster allocation. Staffing is the phase in which predictions of future amount of work are done. The predicted amount of work is then translated into workforce requirements that are necessary to meet the total demand or a certain service level. Shift and roster design are concerned with the creation of shifts and rosters, while shift and roster allocation is related to allocation of (groups of) people to rosters to ensure some staffing level requirements.

In shift design, the shifts depend strongly on the nature of the operations. Other aspects as operation time, regulations (such as breaks, maximum working time, maximum working time before a break), the difficulty of the work and much more, also need to be considered during the design of shifts. Traditionally, the construction of shifts and rosters has been seen as a decision on the tactical level rather than on the operational level. The reason for this is that introducing new shifts/rosters is costly and very time consuming. Nowadays computer-based systems are used to optimize the generation of shifts and the allocation of these shifts to the people. However, these two steps are still performed separately due to operational reasons. As mentioned earlier, the construction of shifts is considered a tactical operation. By changing shifts/rosters too often on the operational level could lead to confusion and unhappiness of the employees. Therefore changes to shifts/rosters shouldn't be done on the daily basis, but rather over a period of time.

Many researchers have studied staff scheduling problems over the years. Therefore there is a lot of literature about staff scheduling. However, there is not much literature about shift design in particular. The first publication on shift design was [13]. Here, the shift design problem was formulated as a set covering problem. First, a column generation approach was used to generate possible shifts. Each column corresponded to a certain shift. Then the set covering problem was solved using a column selection approach. This approach works well for small instances, but it will become impractical if the number of shifts increases. An integer programming formulation for shift scheduling was proposed by Bechtold and Jacobs [4]. The authors obtain better results than the set covering approach in [13]. However, their model is limited to companies that operate less than 24 hours a day. Thompson [25] extends the model from [4] in [25]. Here the length of shifts and the breaks are modeled implicitly. In the last few years, the shift design problem has gotten more interest in the research community. Recent publications such as [21] and [14] show that this problem needs to be studied more extensively.

The rest of this chapter is structured as follows. In the next section, different application areas of staff scheduling will be discussed. Then a brief overview of some solution methods is presented. Although this review of applications and solution methods is about staff scheduling in general and not specifically about shift design problems, it still gives insights on possible applications and solution methods for shift design problems. Finally, this chapter concludes with some recommendations for solving problems in the field of staff scheduling and rostering. Particularly, we focus on the conclusions that were drawn in [10].

2.2 Application areas

Scheduling of staff members happens in many application areas. Whether it's a transportation company, health care or emergency services they all have to schedule their staff members in order to satisfy a certain level of demand or service. The challenge of these applications lies in the dynamic nature of the demand and service requirements, which are often variable over time.

In this section, we'll discuss the application of staff scheduling in several areas. These application areas were taken from [16]. For each area a short description of the scheduling problem and its characteristics are given. Some references on scientific literature for further reading are also included.

2.2.1 Transportation systems

In the transportation market, staff scheduling and rostering is known as crew scheduling and rostering. The transportation market consists of airlines, railways, mass transit and busses. These applications are characterized by spatial and temporal features. This means that each task has a certain starting time and location and also an ending time and location. All the employees perform tasks that are determined by a given timetable e.g. flight, train, subway or bus.

Tasks are the smallest building blocks and can be obtained by decomposing a complete flight, train or bus journey. Examples of tasks are flight legs in airlines and trips between three or more consecutive stops in a bus line.

The most extensively studied application of staff scheduling in the transportation market is the airline scheduling and rostering. The reason for this is because of the magnitude of the business. Managerial decisions on staff levels and scheduling can have a very big impact on the total costs of a company. The airline crew scheduling and rostering problem is most often solved using the well-known decomposition technique. Here, the overall problem is divided into three sub-problems which are solved in different stages. These sub-problems include: (a) crew pairing generation; (b) crew pairing optimization; (c) crew rostering. In the crew pairing generation step, a large number of feasible pairings/duties are generated from the given timetables. In the second step, a selection will be made so that all the flight legs are covered at minimum cost. In the final stage, the selected pairings in the second step will be sequenced into rosters that are then assigned to individual crew members. For references about crew pairing generation, crew pairing optimization and crew rostering, refer to [16].

Another application of staff scheduling in the transport market is crew scheduling and rostering in public transport systems. Similarly to airline crew scheduling and rostering, bus schedules and rosters are constructed using bus timetables. The time scale for bus schedules and rosters are often much smaller than for airlines. The reason for this is that the tasks that need to be performed here can be performed by a crew complement without long rests. The spatial feature also becomes less important than in airline scheduling. This means that it would be preferable to start and end a task at the same location, but this is not a hard constraint in bus crew scheduling and rostering.

Applications of staff scheduling in railways applications have appeared recently in the public transport sector literature. In most cases, the authors report about real applications of crew rostering applications in different countries. For references about bus scheduling and rostering and railway scheduling refer to [16].

2.2.2 Call centers

Staff scheduling in call centers is known as personnel rostering. The main difference between personnel rostering in call centers and crew scheduling in the transportation market is that scheduling in call centers does not involve a spatial feature. This makes the scheduling problem a bit less complex. However, in personnel rostering the exact nature and the number of tasks to be performed are not known beforehand. There exists a certain workforce requirements pattern for the planning horizon, but the structure of this pattern is unknown. Keep in mind that the workforce requirements for call centers vary from day to day and from week to week. For this it is quite challenging to select the right shift start times and lengths in order to obtain good, low-cost rosters that cover the workforce requirements adequately. An extra requirement in personnel rostering is that rosters may need to over cover the demand in certain time intervals and under staff in other time intervals.

The first step in solving the personnel rostering problem is to evaluate the workforce requirements. This can be done by using queuing models and simulation models. Sometimes, these two are also combined to obtain good staff requirements. Once the workforce requirements are known, the next step is to create good shifts and rosters to meet these requirements. Nowadays it's necessary to take into account that not every call is of the same type and therefore different staff with different call-handling skills is also needed. During the creation of the rosters, the skill issue must be considered along with the other usual constraints such as the maximum working time, earliest shift starting time, latest shift end time and so on. The number of feasible shifts that can be created based on these conditions can be very large. The rostering problem is to allocate the pieces of work to individuals with certain call-handling skills. For references about personnel scheduling in call centers, refer to [16].

2.2.3 Health care systems

The most well-known scheduling problem in the domain of health care is the nurse scheduling/rostering problem in hospitals. The rosters that are generated need to take into account that there are enough qualified nurses to satisfy the demand arising from the number of patients. At the same time, these rosters must consider work regulations, distinction between permanent and casual staff, ensuring that night and weekend shifts are distributed fairly, allowing for off days and considering a range of preferences by the employees. The rostering problems that arise in most cases are over-constrained and cannot be solved to optimality. The objective when solving these over-constrained rostering problems, is to find a solution that satisfies all the hard constraints (e.g. staff demand per day with certain skills), while violating the least soft constraints (e.g. employee preferences) as possible.

In the literature, a lot of articles about nurse scheduling and rostering can be found. Some of these articles date back to the 1970's. A recent bibliographic survey on nurse rostering problems is given in [12]. Here, recent models and approaches for the nurse rostering problems are discussed. This paper serves the purpose of giving a brief overview on mathematical programming methods, different kinds of constraint programming techniques and meta-heuristics. A review of automated nurse rostering approaches was presented in [10]. The authors concluded that only a few methods that have been developed in research have been applied on real world data and that even fewer have actually been implemented in real hospital wards.

2.2.4 Protection and emergency services

When staffing police, ambulance, fire and security services, it is important to meet expected service standards. These terms are often specified in terms of response time to attend incidents, the ability to dispatch specified numbers of properly trained people to different types of incidents and so on. Additionally, given the nature of the work, most emergency services have very controlled regulations specifying patterns of work that are acceptable.

One of the key factors in staffing for emergency services is that the frequency of incidents that require emergency people varies at different times of the each day, week or year. An example is all the police and ambulances that are busy during the weekend nights when a lot of people go out or when there is some kind of festival. This variability in the number of staff needed to provide the required service level is what makes this problem so hard.

References on scheduling police officers, ambulance officers and security guards can be found in [16].

2.2.5 Other applications

Other application areas of staff scheduling include: civic services and utilities, venue management, financial services, hospitality and tourism, retail and finally manufacturing. For more details on how staff scheduling is applied in these areas, refer to [16].

2.3 Solution methods

Many solution methods have been applied to staff scheduling problems over the years. In this section, we'll cover some solution methods that have been used for staff scheduling and rostering problems. As could be seen in Section 2.2, there is a lot of literature available on staff scheduling in many different application areas. To give an overview of all these papers is out of the scope of this paper. The purpose of this section is to give a brief overview of solution methods that have been applied on staff scheduling problems. The different solution methods are put in categories similar to the ones presented in [12, 16, 10]. As mentioned in [16], it should be noted that the literature is heavily skewed towards mathematical programming and meta-heuristic approaches.

2.3.1 Mathematical Programming

Mathematical Programming (MP) methods also referred to as exact methods, can be used to find optimal solutions. In MP, scheduling and rostering problems are formulated as Linear Programs (LP), Integer Programs (IP) or some other mathematical program. In both [16, 10], the authors mention three major limitations of why MP is not appropriate for the enormous search spaces that are represented by modern rostering problems. These three limitations are:

- In many cases a column generation method is used and these methods hide much of the complexity of the problem in the definition of the columns. Therefore the sub-problem, the pricing problem, becomes the real challenge and the advantage of using an exact method for the master problem may be lost.

- Constraints and objectives are not easily expressed when using MP formulations. For this reason these approaches are more commonly applied to simplified versions of the real world scheduling problems.

- It takes a lot of time and effort to implement a good MP method for a particular scheduling problem. If the rostering rules and regulations change over time, the implemented model might no longer be useful anymore.

Besides these limitations, a lot of research has been done on solving scheduling problems with MP. Many problems in staff scheduling have been solved using the set covering/partitioning model. A set covering model needs to be solved in both the column generation phase as well as the column selection phase. The most well-known exact algorithm for linear integer programs is branch and bound. Upper bounds are calculated using heuristics and lower bounds can be calculated by using linear programming relaxations and Lagrange relaxations. It is clear that the set covering model represents a dominant approach. It has been employed in over one hundred articles. Other MP methods that have been used in the literature to solve staff scheduling problems are multi-objective approaches, network models, dynamic programming and decomposition techniques. For an overview of these works refer to [12, 16, 10].

2.3.2 Meta-heuristics

When trying to solve large scheduling problems to optimality with exact methods, often a large amount of memory and computation time is required. Sometimes computers can even run out of memory. For these reasons, researchers have turned to heuristics. A heuristic is simply a set of rules that can be applied to a certain problem in order to solve it. These heuristics don't guarantee an optimal solution to the problem. However, they can produce satisfactory results in a reasonably short amount of time. In last decades, meta-heuristics have become increasingly popular to solve large combinatorial optimization problems. Meta-heuristics combines different base methods under one framework. These base methods are often inspired by other disciplines such as classical heuristics, artificial intelligence, biology, neural engineering and statistics. The reason for their increasing popularity is that meta-heuristics have proven to be quite robust and they can generate near optimal solutions with rather short computation times. They are relatively simple to implement and they also make it possible to deal with complex objectives. In the literature there are numerous papers that deal with the application of meta-heuristics to scheduling problems. A few examples of meta-heuristics are Tabu Search (TS), Simulated Annealing (SA), Genetic Algorithms (GA), Ant Colony Optimization (ACO) and Variable Neighborhood Search (VNS). To go into the details of these methods is outside the scope of this thesis. Therefore we advise the reader to refer to [11] and [17] for details and further references on these methods. A short review on some of the applications of meta-heuristics on staff scheduling problems will be given now. This summary is intended to show how meta-heuristics have been applied on different types of scheduling problems such as nurse rostering problem, airline scheduling problem, driver scheduling problem, and others.

Tabu Search

Shen and Kwan presented a tabu search heuristic for the driver scheduling problem in [24]. They designed multi-neighborhoods and memory schemes specifically for driver scheduling problems. Their heuristic has proven to give good results while keeping the computation time very low.

Four tabu search based methods for automatic generation of rotating workforce schedules are proposed in [20]. The proposed methods were implemented and evaluated on some benchmark examples given in the literature and on real test problems. The combination of a min-conflicts heuristic with tabu search has proven to be the most effective. The proposed methods were to be included in a commercial package for automatic generation of rotating workforce schedules.

In [5], the nurse rostering problem is solved using a local search approach. This local search makes use of a greedy procedure to make sure that infeasible solutions can not be generated. Two algorithms are proposed in that paper. One of them is based on tabu search and the other one on iterated local search. Both methods have proven to have good quality solutions and CPU times. The quality of the solutions generated by the greedy algorithm compares well to a straightforward lower bound and they outperform the solutions that are manually generated in the hospital ward.

Simulated Annealing

Brusco and Jacobs used a SA approach to the labor tour scheduling problem in [6]. The objective of this problem is to minimize the number of full-time employees needed to satisfy a forecasted demand. The authors show that by using a SA approach, it's possible to reach good solutions and eliminate the need for long computation times.

A SA approach for developing shift schedules was presented in [26]. The performance of the SA was tested using different search parameters on numerous test problems. The schedules created by the SA approach were slightly more expensive than the optimal schedules, but they were generated in less time.

Emden-Weinert and Proksch [15] present a study of a SA algorithm for the airline crew pairing problem. They combine the SA with a problem specific local improvement heuristic and make use of independent runs.

In [22] Parr and Thompson show how to overcome the problem of setting weights to binding constraints and still being able to find good solutions for the nurse scheduling problem. They compared two methods, namely SAWing and Noising with simulated annealing.

Genetic Algorithms

Burke et al. [7] developed a memetic algorithm where each individual goes through a local improvement before being used for recombination. Infeasible solutions resulting from the recombination phase are repaired using repair procedures. The approach has proven to be robust to instances from the real world. The authors conclude that the memetic approach leads to better solutions, but the compromise is that the computation time is higher. For this, the memetic approach should only be used to create schedules long before they are really needed.

Aickelin [3] presents an indirect GA to a manpower-scheduling problem arising at a major UK hospital. The GA makes use of an indirect coding based on permutations of the nurses and a heuristic decoder that generates schedules from these permutations. The solution space is reduced by applying a hybrid crossover operator. The performance of this GA was compared to a TABU search method.

Ant Colony Optimization

Gutjar and Rauner (2007) [18] were the first to apply ACO to the nurse scheduling problem. They applied their algorithm on a dynamic regional problem which was under discussion at the Vienna hospital compound. The computational results show that their ACO algorithm achieves significant higher improvements compared to a greedy assignment algorithm. Their algorithm helps cover and balance unforeseen peaks of workforce requirements by taking different preferences and costs into account using a cost function.

Variable Neighborhood Search

VNS has been used to solve the nurse rostering problem in [9]. The developed method is able to take a broad variety of constraints into account and it has a greater success rate in escaping from local optima than other previously tested meta-heuristics. The VNS is able to exploit hidden parts of the solution space by applying appropriate problem specific neighborhoods. Both shortsighted neighborhoods as well as greedy ones are used. Their experiments show that good quality schedules can be found in short computation times.

Another paper that applies VNS on the nurse rostering problem is [8]. Here, a hybrid approach that combines heuristic ordering with VNS is presented. The quality of the solutions can be significantly improved by combining heuristic ordering, VNS and backtracking. The proposed method clearly outperforms a commercial GA on small and medium sized instances. However, the GA was still superior on the larger instances.

In both [9] and [8], VNS was applied on the nurse rostering problem. There are also many other applications of VNS on scheduling problems in the literature. However, there is no publication available where VNS has been applied on shift design problems. Since VNS has shown to give good results on other scheduling and rostering problems, it seems like a good candidate solution method for solving the shift design problems that we're dealing with in this thesis.

2.4 Recommendations for designing algorithms

In the 1970's, scheduling problems were primarily solved using exact methods. But for the increasing size of today's scheduling problem instances, even with today's computational power it's not possible to solve these large instances using exact methods. The search space for these large instances is too large and there's a great probability that the program will run out of memory before a solution has been found. The results for these exact methods can be improved by hybridizing them with heuristic approaches. Recent developments show that only some kind of (hybrid) heuristic method offers a realistic way of handling the difficult and challenging problems in the future. Burke et al. [10] state that the current state of the art is represented by interactive approaches that incorporate problem specific methods and heuristics (derived from exploiting specific knowledge of the problem and the constraints) with powerful modern meta-heuristics, constraint based approaches and other search approaches.

Another important recommendation by Burke et al. [10] is that modern nurse scheduling research should develop models and problem solving techniques that capture the needs and requirements of the real world situation. The authors state that a lot of attempts to address nurse scheduling problems have tackled simplified problems. These simplified models do give insights in the solution techniques and they provide benchmark problems to test new algorithms and developments. However, these advances do not contribute to our capability to solve real nurse scheduling problems in real hospitals. Thus there's a big gap between nurse scheduling in theory and practice.

Although the findings of the authors in [10] are primarily about nurse scheduling problems, it can somewhat be generalized to staff scheduling problems as was done by [16]. [10] and [16] outlined some future research directions for the general staff scheduling context. These future research directions will be presented now.

- Multi-criteria reasoning: Staff scheduling in different environments present a range of objectives and requirements. Often these objectives are conflicting. For example, the objective to maximize staff preferences may conflict with the objective that requires a certain number of staff to work a certain shift. Since staff scheduling is inherently multi-objective, the decision support system should also reflect this.

- Flexibility and dynamic reasoning: a planned personnel schedule has to be changed on short notice in order to deal with unforeseen circumstances such as staff sickness and emergencies. Fuzzy methodologies show great potential to address the dynamic nature of the problem in practice and to deal with the uncertainty that is featured in the real world.

- Robustness: a schedule needs to be robust in the sense that it should be able to handle sudden changes. For example, if a schedule that has been created contains one person with a certain expertise and he calls in sick, then the schedule should still contain another person with that same expertise.

- Ease of use: many algorithms that have been developed in the literature need a lot of correct parameters setting in order to generate good solutions. Most often, staff members won't have the knowledge and experience to set those parameters. Therefore it's a significant research goal to investigate algorithmic methods for scheduling problems that are less reliant upon parameters.
- Human/computer interaction: scientific research should focus more on the interfaces between human input and the scheduling process in order to overcome the issue of only solving simple models of scheduling problems.
- Problem decomposition: the idea of breaking a large scheduling problem into smaller and easier to handle sub-problems has shown great potential in recent works. The only drawback of this method is that it's possible to make allocations in earlier sub-problems. This means that it's possible that the later sub-problems become infeasible. Solving this issue will make it possible to solve larger instances of scheduling problems.
- Exploitation of problem specific knowledge: more constraints and requirements of real problems should be investigated. It might be possible to exploit certain knowledge about solutions that can be used in order to improve the performance of a search algorithm.
- Hybridization: the best way to improve the current solution algorithms is to draw on the strengths and capabilities of the existing methods. By combining different methods, it should be possible to create a new method that is better than its two successors.
- Inter-disciplinary: it is safe to say that in order to achieve all the research themes that have been mentioned will require research expertise from other disciplines such as operations research, artificial intelligence, healthcare, administration, management, psychology and software engineering.

With these recommendations in mind, our goal is to design an algorithm for the shift design problem that satisfies as many of these recommendations as possible. The proposed algorithm will be integrated in the TNT Express manpower planning system (MPT). This algorithm is going to be used primarily to solve real world instances of shift design problems and to create schedules on the tactical level. In the next chapter, a formal description and mathematical formulation of the minimum shift design problem will be given.

Chapter 3

Minimum Shift Design Problem

3.1 Background

In order to create a complete workforce schedule, there are a few steps that need to be taken. First, an estimation of the total workload that is required for each time slot in the planning horizon needs to be made. Then shifts need to be designed, the number of employees that are needed for each shift must be determined and finally the employees must be assigned to the designed shifts. As mentioned in Chapter 1, there are two main approaches to creating a complete workforce schedule. The first approach tries to solve the design of shifts and the assignment of shifts to the employees as one single problem. The other approach is to design the shifts first, determine the number of employees needed for each shift and then schedule the actual employees afterwards. This thesis focuses on the latter approach. We focus on the design of shifts and determining the number of employees to be assigned to each designed shift. This problem is known in the literature as the Minimum Shift Design problem (MSD). The MSD problem was first introduced by Musliu et al. in [21] and it has been further studied in [14]. This chapter will go into the details of the MSD problem.

The rest of this chapter is structured as follows. In the next section, the formal definition of the MSD problem will be given. Examples will be used to illustrate elements of the MSD problem. Then we'll discuss the methods that have been used in the literature to solve this problme. Finally, we'll propose a new solution method for this problem.

3.2 Problem definition

The MSD problem consists of the selection of which shifts to use and how many people to assign to each such shift in order to satisfy pre-specified workforce

Start	End	Mon	Tue	Wed	Thu	Fri	Sat	Sun
00:00	06:00	5	5	5	5	5	5	5
06:00	08:00	2	2	2	6	2	0	0
08:00	09:00	5	5	5	9	5	3	3
09:00	10:00	7	7	7	13	7	5	5
10:00	11:00	9	9	9	15	9	7	7
11:00	14:00	7	7	7	13	7	5	5
14:00	16:00	10	9	7	9	10	5	5
16:00	17:00	7	6	4	6	7	2	2
17:00	22:00	5	4	2	2	5	0	0
22:00	24:00	5	5	5	5	5	5	5

Table 3.1: Example of a planning horizon with workforce requirements taken from [21]. The columns indicate the number of employees needed for each hour in the time period specified in the rows. Example: (00:00-06:00, Mon) = 5 means that for each hour on Monday between 00:00 and 06:00 5 employees are required.

requirements. This section gives the formal definition of this problem. The formulation that is given here was taken from [21] and [14].

Planning horizon with workforce requirements

The planning horizon for the MSD problem is given in d planning days, $d \in D$ where $D = 1, \ldots, d$. Each planning day starts at a certain specified time on the regular day and it ends 24 hours later, usually on the next calendar day. Each planning day j consists of n equal-size smaller intervals $t_i = [\tau_i, \tau_{i+1})$, called timeslots. Each timeslot has length $h = \| \tau_{i+1} - \tau_i \| \in \mathbf{R}$ expressed in minutes. The planning horizon starts on time point τ_1 on the first planning day and ends at time point τ_{n+1} on the last planning day. Every timeslot t_i of day j contains *an expected number of employees* b_{ij}. This number represents the number of employees that should be present during that timeslot. An example of a planning horizon with workforce requirements can be seen in Table 3.1. This example was taken from [14].

Shifts

The goal when solving the MSD problem is to find a set of *shifts* that covers all the workforce requirements. Each shift is characterized by two values, namely the starting time indicated by σ_s and the length of the shift indicated by λ_s. The set of all possible shifts is denoted by S. A shift can be denoted as $s = [\sigma_s, \sigma_s + \lambda_s)$. Because the timeslots that are being used are discrete, the variables σ_s for each shift s can only assume values of τ_i and the variables λ_s can only be a multiple of the timeslot length h.

Shift type	min_s	max_s	min_l	max_l
M (morning)	05:00	08:00	07:00	09:00
D (day)	09:00	11:00	07:00	09:00
A (afternoon)	13:00	15:00	07:00	09:00
N (night)	21:00	23:00	07:00	09:00

Table 3.2: Example of shift types with different starting times and lengths taken from [21].

Shift types

There are several constraints that make the problem of designing shifts more complex. Two of these constraints are concerned with the starting time of a shift and also the duration of a shift. It is quite straightforward that not all starting times are allowed and that shifts should have a maximum length. The designed shifts must satisfy these constraints in order to be considered feasible. To deal with these constraints, the problem definition also includes a set of *shift types* $V = v_1, \ldots, v_r$. Each shift type is characterized by the *earliest* and the *latest starting times* denoted by $min_s(v_k)$ and $max_s(v_k)$ respectively, and a *minimum* and *maximum length* of its shifts denoted by $min_l(v_k)$ and $max_l(v_k)$. Each shift s must belong to one unique shift type. This means that its starting time and length must lie within the intervals that are defined for that particular shift type. The shift type relation can be denoted with $K(s) = v_k$. This relation states that a shift s that belongs to the type v_k is a feasible shift if $min_s(v_k) \leq \sigma_s \leq max_s(v_k)$ and $min_l(v_k) \leq \lambda_s \leq max_l(v_k)$. Table 3.2 shows an example of different shift types. Here each shift type has its own allowed starting times and lengths. This example was taken from [14].

Objective

The objective of the MSD problem is to select a set of q feasible shifts $Q = \{s_1, s_2, \ldots, s_q\}$ and to determine the number of employees $x_j(s) \in N$ that needs to be assigned to each shift $s \in S$ for each day j in order to satisfy the workforce requirements b_{ij} at each timeslot t_i of the day. By denoting the collection of shifts that are in the timeslot t_i with $S_{t_i} \subseteq Q$, we can state that a feasible solution consists of d numbers of $x_j(s)$ assigned to each shift s such that $l_{ij} = \sum_{s \in S_{t_i}} x_j(s)$ is equal to b_{ij}. This means that for all values of i and for all values of j, the workforce requirements must be satisfied.

Optimize three objectives

For most real instances, the constraint of satisfying all workforce requirements is relaxed and small deviations are allowed. Because of this, it's common practice to evaluate the solutions for the MSD problem using an objective function. The objective function is a weighted sum of three main components and it needs to be minimized. The first component is the sum of the excess of workers (or

Start	Type	Length	Mon	Tue	Wed	Thu	Fri	Sat	Sun
06:00	M	08:00	2	2	2	6	2	0	0
08:00	M	08:00	3	3	3	3	3	3	3
09:00	D	08:00	2	2	2	4	2	3	2
14:00	A	08:00	5	4	2	2	5	0	0
22:00	N	08:00	5	5	5	5	5	5	5

Table 3.3: Example of a solution for the problem in Table 3.1 taken from [21].

overcover) during the planning horizon and it can be denoted as $F_1(Q, x) = \sum_{j=1}^{d} \sum_{i=1}^{n} max\{l_{ij} - b_{ij}, 0\}$. The reason for taking the *max* for the excesses is that when $l_{ij} - b_{ij} < 0$, it means that there is a shortage of employees. The shortages are counted in the second component, so it suffices to set $F_1(Q, x) = 0$ for this time period. The same principle applies for the second component. The second component is the sum of the shortages of workers (or undercover) during the planning horizon. This component can be denoted as $F_2(Q, x) = \sum_{j=1}^{d} \sum_{i=1}^{n} max\{b_{ij} - l_{ij}, 0\}$. The *max* is also used here because the excesses are counted in the first component. Finally, the third component represents the number of shifts that have been selected in this solution. This can be denoted as $F_3(Q, x) = \mid Q \mid$.

The MSD problem can be modeled as an optimization problem with three objectives where each criterion has a certain importance depending on the situation. Therefore, different weight factors can be assigned to the three F_c components. Thus the people using a shift design system are able to adjust the weights according to their preferences.

An example of a solution for the MSD problem described in Table 3.1 and Table 3.2 is given in Table 3.3. This solution is shown graphically in Figure 3.1. This solution is not optimal, because in the figure there are periods with shortages and periods with excesses. In the shortage periods, the area under the required line is not filled. In the excess periods, there is a colored area on top of the required line. The value of the objective F_1 is 15 man-hours of shortage, the value for F_2 is 7 man-hours of excess and the number of shifts used, F_3, is 5 (see Table 3.3).

3.3 Solving the mathematical model

Musliu et al [21] solved the MSD problem using a local search approach. They proposed basic and composite moves for generating the neighborhood in every iteration. In order to prevent cycles during the search, they used principles of tabu search and they also used other mechanisms to prohibit the previous solutions to be descendant of the current solution. Additionally, they proposed a procedure to generate a good initial solution and a method which exploits domain knowledge about the problem during the search. The authors acknowledge in the conclusion of that paper that the local search can be further refined.

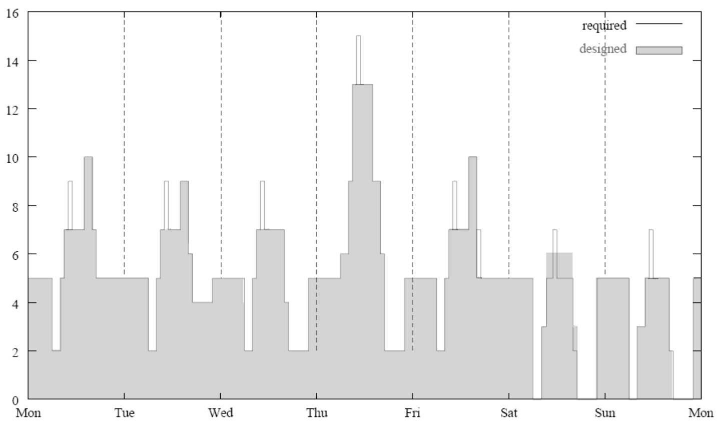

Figure 3.1: Solution to the example taken from [21]

Di Gaspero et al [14] used a hybrid heuristic algorithm composed of a greedy constructive heuristic and a multi-neighborhood tabu local search to solve the MSD problem. The authors show that an important source of hardness is related to the variability in the size of the solution. By dropping the requirement of minimizing the number of shifts used, the MSD problem becomes solvable in polynomial time. Their experimental results show that their hybrid heuristic algorithm outperforms the local search approach in [21]. And thus they conclude that their hybrid heuristic can be considered as the best general-purpose solver among the other heuristics that were compared to it.

The literature overview in Chapter 2 showed that there are two main streams of solution methods for complex optimization problems. These main streams are MP and meta-heuristics. Some of the advantages and disadvantages of these methods were also presented. It can be concluded that MP methods are too computationally expensive and they're often problem specific. For these reasons, our choice for a solution method has fallen on heuristics. Further motivation for this choice is that heuristics are relatively simple to implement and they have shown to be robust to different instances of scheduling problems.

In the next chapter, our solution method for the MSD will be presented. This solution method is based on the principles of the meta-heuristic Variable Neighborhood Search. More details on our motivation to choose this approach and a complete description of our algorithm will also be given.

Chapter 4

Variable Neighborhood Search

4.1 Introduction

Local search is a well-known and general approach to solve hard combinatorial optimization problems. Usually the goal of these problems is to find an optimal solution that has the minimum (or maximum) cost. In order to apply a local search heuristic on an optimization problem, one must define the so called *neighborhood structures* for the solutions. The local search starts from an initial solution (e.g. randomly created) and from there it keeps moving to a better neighbor as long as that's possible. Finally, it terminates at a *locally optimal* solution, one that does not have a better neighbor.

This framework has been applied successfully to many optimization problems. There are two important issues that need to be taken into account when using local search algorithms: (1) the quality of the obtained solutions and (2) the complexity of the local search heuristic. The first issue concerns the quality of the local optima for the different neighborhood structures and their relationship with the global optimum. The second issue is concerned with how fast the algorithm can reach local optima. A good local search heuristic should have a good balance between the two. It must be able to find the global optimum using the defined neighborhood structures and do this in a relatively short amount of time [1].

The challenge for local search algorithms lies in finding the global solution in the search space. Most local search algorithms, as its name implies, will find the local optimum. But the local optimum to an optimization problem is not likely to be the global optimal solution. The local optimal is often very dependent of the initial solution where the search algorithm started at. To prevent the local search algorithm to be trapped in a local optimum, different techniques can be used. One of the methods that can be applied is to multi-start the local search. Here, the local search algorithm doesn't start from one point, but it will have different starting points in the search space. Each initial solution will result in a different local optimum. The best solution of these local optima, can be seen as the global optimum.

In the past decades, meta-heuristics such as GA, SA, TABU search and Variable Neighborhood Search (VNS) have shown to find near global solutions to optimization problems [11]. Most of these meta-heuristics are an extension of local search techniques, but they have the ability to escape from a local optima. This is what makes them more powerful than a simple local search. In GA for example, escaping local optima is done by not applying only one crossover operator, but also to do an extra operation called mutation on the candidate solution. This mutation operator adds diversity to the candidate solutions, making it possible to explore more places in the search space. Also, by applying elitism with a certain probability eliminates the greediness of the algorithm. By accepting solutions of less quality keeps the search space more diverse and thus enables the GA to explore larger parts of the search space.

In SA, not only solutions that have an improvement in objective value are accepted. Solutions that cause deteriorations are also accepted to a limited extent. In the beginning of SA, large deteriorations are accepted. As the search progresses, smaller deteriorations will be accepted until finally no more deteriorations are accepted. Because of this feature of SA, it can escape from local minima. The basic principles of TABU search is to continue the search from a local optimum by allowing non-improving moves. Certain moves that were executed in previous iterations are prohibited by making them TABU. The TABU mechanism prevents cyclic searches through the same area of the search space. This gives the algorithm the ability to escape from local minima.

Variable Neighborhood Search is a meta-heuristic that escapes from local optima by searching through all the neighborhood areas of a candidate solution. In the next iteration, the search is not continued from the last point the algorithm stopped. But the search starts at a neighbor of the candidate solution. This results in a more thorough search by the algorithm. Since VNS has successfully been applied in the field of scheduling and rostering, our choice has also fallen on this meta-heuristic. Other reasons why this algorithm has been chosen is that it is quite easy to implement and it doesn't require a lot of parameter settings when used. Also, using VNS will enable us to continue researching with some of the framework that was built by Musliu, Di Gaspero et al. An example of this framework would be the neighborhood structures that they have defined. These structures can easily be applied in a VNS algorithm. Finally, VNS has also proven to be able to find optimal solutions in very short computation times.

The rest of this chapter presents our search algorithm for the MSD problem. This search algorithm is based on the principles of Variable Neighborhood Search (VNS). In the next section, a broad description of this meta-heuristic will be given. Subsequently, the objective value that can be used to discriminate between solutions is introduced. Then the neighborhood structures of our search algorithm are defined. Following that, the algorithm to create an initial solution is presented. A couple of processing steps to add diversity and remove unused or similar shifts are discussed. Then, an overview of our search algorithm is given. Finally, an extension to the proposed algorithm is also presented.

4.2 Variable Neighborhood Search

Variable Neighborhood Search (VNS) is a meta-heuristic that was first introduced by Mladenovic and Hansen in 1997 [19]. VNS exploits systematically the idea of neighborhood changes with the goal of finding the local optima's and escaping the valleys that contain them. The algorithm starts the descent at a certain starting point and reaches a local minimum. Then, a series of different predefined neighborhoods of this solution are explored. Here, one or several points of the current neighborhood are used as starting points to run a local descent method and the algorithm stops at a local minimum. The search will jump to this new local minimum if and only if it is better than the incumbent [11].

One of the major differences between VNS and other meta-heuristics is that the basic schemes of VNS are simple and the algorithm requires very few parameters. In some cases even no parameters are needed at all. Therefore, not only does VNS provide very good solutions, but it is also much simpler than other methods. VNS exploits systematically the following observations [11]:

1. A local minimum with respect to one neighborhood structure is not necessary so for another.

2. A global minimum is a local minimum with respect to all possible neighborhood structures.

3. For many problems local minima with respect to one or several neighborhoods are relatively close to each other.

The last observation, which is empirically proven, implies that a local optimum often provides some information about the global optimum. It is possible that several variables have the same value as this local optima. However, it is usually not known which variables should have the same value. Therefore, a structured search of the neighborhood of this local optimum is performed until a better one is found. The rest of this section will give a description of the VNS that is based on [11].

Variable Neighborhood Descent

Variable Neighborhood Descent (VND) can be considered the deterministic version of VNS. It is based on *observation 1* mentioned above, i.e., *a local optimum for a first type of move $x \leftarrow x'$ (within neighborhood $N_1(x)$) is not necessary one for another type of move $x \leftarrow \tilde{x}$ (within neighborhood $N_2(x)$).* Thus it might be advantageous to combine descent heuristics. The basic VND scheme is as follows:

Initialization. Select the set of neighborhood structures N_ℓ, for $\ell = 1, \dots, \ell_{\max}$, that will be used in the descent; find an initial solution x (or apply the rules to a given x);

Repeat the following sequence until no improvement is obtained:

1. Set $\ell \leftarrow 1$;
2. *Repeat* the following steps until $\ell = \ell_{\max}$:
   - *Exploration of neighborhood*. Find the best neighbor x' of x ($x' \in N_\ell(x)$);
   - *Move or not*. If the solution x' thus obtained is better than x, set $x \leftarrow x'$ and $\ell \leftarrow 1$; otherwise, set $\ell \leftarrow \ell + 1$;

4.2.1 Reduced Variable Neighborhood Search

The stochastic version of the VNS is known as the Reduced Variable Neighborhood Search (RVNS). In RVNS, solutions from the pre-selected neighborhoods are chosen at random, without being followed by descent. The efficiency of this method is based on *observation 3* described above. A set of neighborhoods $N_1(x), N_2(x), \dots, N_{k\max}(x)$ will be considered around the current point x (which can be a local optimum). Usually, these neighborhoods are nested, in other words each one contains the previous one. Then a point is randomly chosen in the first neighborhood. If the value for this point is better than that of the incumbent (i.e. $f(x') < f(x)$), the search will be moved there ($x \leftarrow x'$). Otherwise, continue the search from the next neighborhood. After all neighborhoods have been considered, begin again at the first until a stopping condition is met (e.g. max number of iterations or maximum computation time). The RVNS scheme is as follows:

Initialization. Select the set of neighborhood structures N_k, for $k = 1, \dots, k_{\max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

Repeat the following steps until the stopping condition is met:

1. Set $k \leftarrow 1$;
2. *Repeat* the following steps until $k = k_{\max}$:
   - *Shaking*. Generate a point x' at random from the kth neighborhood of x ($x' \in N_k(x)$);
   - *Move or not*. If this point is better than the incumbent, move there ($x \leftarrow x'$), and continue the search with $N_1(k \leftarrow 1)$; otherwise, set $k \leftarrow k + 1$;

4.2.2 General Variable Neighborhood Search

Combining the schemes for VND and RVNS results in the scheme for the General Variable Neighborhood Search (GVNS). This scheme is as follows:

> *Initialization.* Select the set of neighborhood structures N_k, for $k = 1, \dots, k_{\max}$, that will be used in the shaking phase, and the set of neighborhood structures N_ℓ, for $\ell = 1, \dots, \ell_{\max}$, that will be used in the local search; find an initial solution x and improve it by using RVNS; choose a stopping condition;
>
> *Repeat* the following steps until the stopping condition is met:
>
> 1. Set $k \leftarrow 1$;
> 2. *Repeat* the following steps until $k = k_{\max}$:
>   - *Shaking*. Generate a point x' at random from the kth neighborhood of x ($x' \in N_k(x)$);
>   - *Local search by VND*.
>   - Set $\ell \leftarrow 1$;
>   - *Repeat* the following steps until $\ell = \ell_{\max}$:
>   - *Exploration of neighborhood.* Find the best neighbor x'' of x' in $N_\ell(x')$;
>   - *Move or not.* If $f(x'') < f(x')$ set $x' \leftarrow x''$ and $\ell \leftarrow 1$; otherwise set $\ell \leftarrow \ell + 1$;
>   - *Move or not*. If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $N_1(k \leftarrow 1)$; otherwise, set $k \leftarrow k + 1$;

In this scheme, a series of neighborhood structures that define the neighborhoods around any point $x \in X$ of the solution space are first chosen. Then an initial solution is created and improved using RVNS. Afterwards, the VND is used to find a local optimum x. Then a point x' is selected at random within the first neighborhood $N_1(x)$ of x and another descent using VND is done from x'. This results in a new local minimum x''. At this stage, there are three possible outcomes: (1) $x'' = x$, this means that the search is again at the bottom of the same valley; if this is the case, then the search is continued using the next neighborhood $N_k(x), k \geq 2$; (2) $x'' \neq x$ but $f(x'') \geq f(x)$, this means that another local optimum has been found that is not better than the incumbent; in this case too the search is continued using the next neighborhood; (3) $x'' \neq x$ and $f(x'') < f(x)$, this means that a local optimum better than the incumbent has been found; in this case the search is recentered around x'' and begins again with the first neighborhood. When the last neighborhood is reached without finding a solution better than the incumbent, the search begins again at the first neighborhood $N_1(x)$ until a stopping condition is satisfied.

Start	Type	Length	Mon	Tue	Wed	Thu	Fri	Sat	Sun
06:00	M	08:00	2	2	2	6	2	0	0
08:00	M	08:00	3	3	3	3	3	3	3
09:00	D	08:00	2	2	2	4	2	3	2
14:00	A	08:00	5	4	2	2	5	0	0
22:00	N	08:00	5	5	5	5	5	5	5

Table 4.1: Example of a solution for the MSD problem taken from [21].

4.3 Fitness function

In order to be able to measure the quality of different solutions, a fitness function (or objective function) needs to be defined. As mentioned in Chapter 3.2, the MSD is an optimization problem with three objectives. The goal of this problem is to minimize the following three components (1) the excess of workers, (2) the shortage of workers and (3) the number of shifts while satisfying some constraints. The sum of the excess of workers during the planning horizon can be denoted as $F_1(Q,x) = \sum_{j=1}^{d}\sum_{i=1}^{n} max\{l_{ij} - b_{ij}, 0\}$ and the sum of the shortages of workers during the planning horizon can be denoted as $F_2(Q,x) = \sum_{j=1}^{d}\sum_{i=1}^{n} max\{b_{ij} - l_{ij}, 0\}$. Finally, the number of shifts that have been selected can be denoted as $F_3(Q,x) = \mid Q \mid$. In this notation, Q is a set of feasible shifts, b_{ij} are the workforce requirements and l_{ij} are the numbers of employees assigned to a given timeslot i on planning day j. A weight factor w_c can be assigned to each of the three F_c components. Thus, the fitness function for our VNS algorithm is as follows: $w_1F_1(Q,x) + w_2F_2(Q,x) + w_3F_3(Q,x)$.

4.4 Neighborhood structures

A solution to the MSD problem consists of shift times and the number of employees assigned to each shift. An example of a solution S_1 is shown in Table 4.1 (this is the same as Table 3.3). A neighborhood of a solution S is a solution that is in the same neighbor as S, but it's slightly different. Take the simple case where solution S_1 has only one shift from 09:00 till 17:00 and the number of employees for this shift over the week is [4, 4, 4, 4, 6, 6, 2]. A neighbor of this solution could be a solution S_2 that has the same shift times but another assignment of employees i.e. [4, 4, 4, 3, 6, 6, 2]. Solutions S_1 and S_2 are almost identical, the only difference between them is the number of employees assigned to the fourth time slot. S_1 has 4 employees assigned here, whereas S_2 has 3 employees assigned. S_1 and S_2 are thus considered to be each others neighbors in the search space.

For each optimization problem, the solutions are different from each other based on the neighborhood structures. For the Traveling Salesman Problem (TSP), where the goal is to find a round trip through several cities while minimizing the total distance traveled, the neighborhood structures are different from the ones for our shift design problem. Choosing the right neighborhood structures is key to finding good solutions to the optimization problem. The neighborhood structures that we used in this thesis are the same as the ones in [14]. Here, three neighborhood structures were defined: 1) Change Staff (CS), 2) Exchange Staff (ES) and 3) Resize Shift (RS). Each neighborhood structure will be discussed in more details now.

Change Staff (CS) In this structure a new neighbor is created by adding or subtracting one employee for any of the shifts in the solution. The staff of a shift s_i becomes either $x'_j(s_i) := x_j(s_i) + 1$, or $x'_j(s_i) := x_j(s_i) - 1$.

Exchange Staff (ES) Using this structure means that two shifts will be selected. From these two shifts, one employee will be subtracted from one of the shifts and an employee will be added to the other shift. In other words, one employee for a given day is moved from a shift s_{i1} to another one of the same type (i.e. s_{i2}): $x'_j(s_{i1}) := x_j(s_{i1}) - 1$ and $x'_j(s_{i2}) := x_j(s_{i2}) + 1$

Resize Shift (RS) The last neighborhood structure is used to adjust the length of the shifts. It is possible to make a shift shorter by making the start time later or by making the end time earlier. The other possibility is to make a shift longer. This can be achieved by making the start time earlier or by making the end time later. Keep in mind that all newly created shifts must be feasible with respect to the shift type $\mathrm{K}(s_i)$. Let's denote the change in length to a shift s_i as $\delta = +1$ when the shift is enlarged by one timeslot and $\delta = -1$ when the shift is shrunk by one timeslot. If the adjustment is made on the left-hand side of the s_i, we have $\sigma'_i := \sigma_i - \delta h$ and $\lambda'_i := \lambda_i + \delta h$. Should the move take place on the right-hand side, σ_i remains unchanged and $\lambda'_i := \lambda_i + \delta h$.

4.5 Initial solution

The initial solution used in the search algorithm is created randomly. The reason for this is that we want to start with an initial solution that is very diverse. By using a very diverse initial solution will help in exploring larger parts of the search space. Our initial random solution will be created as follows. Five shifts are created for each shift type. The start time, end time and length of the shifts are determined randomly within the constraints of the shift type $\mathrm{K}(s_i)$. Finally, each shift also gets a random number of employees varying between 0 and 3. The motivation for these numbers will be discussed in our experimental results.

4.6 Processing steps

During the optimization, it is likely that shifts that are already present in the solution will be created again and that certain shifts no longer have any employees assigned to them. Both of these situations are not desirable, because they can have a negative impact on the efficiency of the algorithm. Therefore two extra processing steps are added to our proposed algorithm: delete shifts and insert shifts. These processing steps are performed after each iteration of the GVNS.

4.6.1 Delete shifts

The delete shifts processing step consists of two different methods. The first method is used to remove inactive shifts and the second method is used to remove identical shifts.

Remove inactive shifts An inactive shift can be defined as a shift without any workers assigned to it, e.g. shift 1 starts from 08:00-17:00 with zero workers in each time slot (0,0,0,0,0,0,0). During the remove inactive shifts processing step, all shifts that don't have any workers assigned to them will be removed from the current solution.

Remove identical shifts In this method, all shifts that are the same in the current solution are merged together to form one new shift. All the number of workers for each time slot of the shifts are summed and this is the number of workers for each time slot in the new shift. An example of this is step will be given now. Shift 1 starts from 12:00-20:00 and has the following number of workers for each time slot (1,1,1,1,2,2,1). Shift 2 starts from 12:00-20:00 and has the following number of workers for each time slot (3,3,3,3,3,3,3). By merging these two shifts together, a new shift starting from 12:00-20:00 is created. The number of workers for the time slots are (4,4,4,4,5,5,4).

4.6.2 Insert shifts

To make sure that enough diversity remains during the search, new shifts will be inserted during each iteration of the GVNS after the inactive shifts have been removed. These shifts are created the same way as the initial solution. The only difference is that only one shift will be created here for each shift type. Also, the newly inserted shifts don't have any workers assigned to them. This way, they will not have an effect on the overall objective value of the current solution. The reason for this is that the objective value is based on shifts that have workers assigned to them. So if a solution contains 10 shifts and only 5 of them have workers assigned, then only 5 shifts will contribute to the objective value. This results in more diversity during the search without reducing the quality of the current solution.

4.7 Overview of proposed VNS algorithm

The outline of our proposed search algorithm is follows.

Initialization. The set of neighborhood structures for the shaking phase, RVNS and the local search consists of N_1=ChangeStaff, N_2=ExchangeStaff and N_3=ResizeShift. The initial solution x is created randomly and this is improved using RVNS. There are two stopping conditions. The first is the number of iterations for the GVNS and the second is the number of iterations without improvements during the GVNS.

Repeat the following sequence until the stopping condition is met:

1. *Delete* inactive shifts of the current solution.
2. *Insert* new shifts to the current solution.
3. *Delete* identical shifts of the current solution.
4. Set $k \leftarrow 1$;
5. *Repeat* the following steps until $k = k_{\max}$:
    - *Shaking*. Generate a point x' at random from the kth neighborhood of x ($x' \in N_k(x)$);
    - *Local search*. Apply local search using VND.
        - Set $\ell \leftarrow 1$;
        - *Repeat* the following steps until $\ell = \ell_{\max}$:
            - *Exploration of neighborhood.* Find the best neighbor x'' of x' in $N_\ell(x')$;
            - *Move or not.* If $f(x'') < f(x')$ set $x' \leftarrow x''$ and $\ell \leftarrow 1$; otherwise set $\ell \leftarrow \ell + 1$;
    - *Move or not*. If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $N_1 (k \leftarrow 1)$; otherwise, set $k \leftarrow k + 1$;

4.8 Population Based VNS

4.8.1 Motivation

In the preliminary experimental phase, we discovered that the VNS algorithm performs well, but it is not able to generate near optimal solutions for the test examples. To try and overcome this issue, we wanted to extend our proposed VNS algorithm. At the moment, the main drawback of VNS algorithm is that it's based on a local search method. This means that only a single configuration is kept at every iteration. From our preliminary experiments, we can conclude that this is not good enough. Thus we have to turn to methods that have several configurations in the same iteration. These are the so called *population-based* search algorithms. The availability of several configurations at the same time, allows the use of more powerful mechanisms to traverse through the search space. Some of the most well-known population-based methods are Evolutionary Algorithms (EAs). EAs are randomized search methods that can provide optimal or near optimal solutions for combinatorial optimization problems. EAs are inspired by the natural selection and evolution processes. Here the natural principles, such as survival of the fittest, are applied. Ever since their introduction, EAs have been popular because they can contribute to finding good solutions for complex mathematical problems. The most popular EAs are Genetic Algorithm (GA), Genetic Programming (GP) and Evolutionary Strategy (ES) [11]. Since EAs have been applied so successfully in the last decades, it is our strong belief that by transforming a local search VNS into a population-based VNS should improve really the performance of our proposed algorithm. In the rest of this section, we'll present an extension of our proposed VNS algorithm called Population Based Variable Neighborhood Search (PBVNS).

4.8.2 Modifications

The main difference between PBVNS and VNS is that unlike our previous VNS algorithm that uses one solution at a time, the PBVNS will go through the search space using a population of solutions. This will help in escaping local optima and exploring more parts of the search space. In order to create this new population-based method, we'll have to modify our VNS algorithm. These modifications will be discussed now.

Initial population Since we're no longer working with one single solution, we'll have to generate a population P of random initial solutions. Each solution x_p is still a set of feasible shifts with numbers of employees assigned to each shift. The way each individual solution x_p is created, remains the same as in the VNS algorithm. Five shifts are created for each shift type K. The start time, end time and length of the shifts are determined randomly within the constraints of the shift type $K(s_i)$. Finally, each shift also gets a random number of employees varying between 0 and 3.

RVNS with population In GVNS, the RVNS is used to improve the initial solution before applying the local search part of the algorithm. This way, the search will already be centered around a good area when the VND begins. Since the RVNS has proven to play an essential in the efficiency of the GVNS, we'll also use this part of in our PBVNS. This means that all the solutions in the initial population are first randomly generated and then improved using RVNS.

Shake and Local Search After the population of solutions has been generated and improved, the PBVNS goes into the main part of the algorithm. Here, the PBVNS takes a neighbor x'_p from each solution x_p (shake) and starts the VND from each of those points. When no more improvements can be found with the VND, we will have found a new solution x''_p for each solution x_p in population P.

Update population One of the most important questions in population-based algorithms is how to update the population with the newly created solutions. In most population-based algorithms, the population size cannot be modified when the algorithm is running. This means that some solutions in the current population will have to be discarded in order to make place for the new solutions. This step is key in the performance of population-based algorithms.

Traditionally, there are two approaches to update a population. These two approaches are taken from the literature on EAs. The first approach is the so called *basic generational replacement*. Here, the entire old population is replaced with the newly generated solutions. The hope is that the new solutions have carried all the necessary information to the next generation. However, the drawback of this method is that the best solutions in the population do not go to the next generation. Also, it could be the case that certain important information was lost during the optimization phase in this generation. Therefore the next generation will miss this important information. An alternative to this approach is the so called *stead state replacement*. Here, new solutions are inserted into the population as soon as they have been created. As opposed to replacing an entire population at once, the stead stay state replacement only replaces some individuals of the population. Most often, the solution that is replaced is either the worst in the population or simply chosen randomly.

In our PBVNS, we'll make use of the stead state replacement. If a newly found solution x''_p after VND is better than any x_p solution in P, then the worst x_p is replaced with x''_p with a certain probability. The reason for using this probability is to make sure that the PBVNS doesn't become a greedy algorithm. Greedy optimization algorithms that only accept improvements during the search, are prone to converge prematurely and get stuck in a local optimum. By giving the bad solutions a chance to go to the next generation, adds diversity to the solutions in P. Eventually, this will only have a positive effect on the performance of the PBVNS. After some preliminary experimenting, this probability was set to 0.75 and it doesn't change when the algorithm is running.

4.8.3 Outline of PBVNS

By applying all the modifications that were just mentioned, we extended our proposed VNS algorithm to the PBVNS algorithm. The following scheme shows our proposed PBVNS algorithm for the MSD problem:

Initialization. The set of neighborhood structures for the shaking phase, RVNS and the local search consists of N_1=ChangeStaff, N_2=ExchangeStaff and N_3=ResizeShift. The initial population P is created randomly and each solution x_p is improved using RVNS. There are two stopping conditions. The first is maximum number of iterations for the PBVNS and the second is the number of iterations without improvements during the PBVNS.

Repeat the following sequence until the stopping condition is met:

1. _Delete_ inactive shifts of each solution x_p.
2. _Insert_ new shifts to each solution x_p.
3. _Delete_ identical shifts of each solution x_p.
4. Set $k \leftarrow 1$;
5. _Repeat_ the following steps for each solution x_p until $k = k_{\max}$:
    - _Shaking_. Generate a point x'_p at random from the kth neighborhood of x_p ($x'_p \in N_k(x_p)$);
    - _Local search_. Apply local search using VND.
        - Set $\ell \leftarrow 1$;
        - _Repeat_ the following steps until $\ell = \ell_{\max}$:
            - _Exploration of neighborhood._ Find the best neighbor x''_p of x'_p in $N_\ell(x'_p)$;
            - _Move or not._ If $f(x''_p) < f(x'_p)$ set $x'_p \leftarrow x''_p$ and $\ell \leftarrow 1$; otherwise set $\ell \leftarrow \ell + 1$;
    - _Move or not_. If this local optimum x''_p is better than x_p, move there ($x_p \leftarrow x''_p$), and continue the search with $N_1(k \leftarrow 1)$; otherwise, set $k \leftarrow k + 1$;
6. _Update population_. For each newly generated local optimum x''_p, check whether it's better than any x_p in P. If it's better, then replace x_p with x''_p in P.

The next chapter presents the experiments and computational results of our proposed algorithms on the test cases that were used in both [21] and [14]. The proposed algorithm will also be applied on a real world case of the shift design problem.

Chapter 5

Experiments and computational results

This chapter presents the computational results of our experiments. The results are obtained using our implementation of the proposed solution method in Borland Development Studio (BDS) Delphi 2006. The choice for using BDS was straightforward, as the manpower planning system at TNT Express was built using BDS. By implementing our algorithm in BDS facilitates the integration between the proposed algorithm and the manpower planning system.

This chapter is outlined as follows. First, our experimental setup containing the performance measures for the evaluation of the proposed algorithm, a description of the test data and details of the experiments are introduced. Afterwards, we'll present a comparison between the performance of the proposed VNS algorithm and the PBVNS. Next, the experimental results for the PBVNS are discussed. These results are used to illustrate the best set of parameters for the PBVNS. Then, the results of our proposed algorithm using the best set of parameters is presented and they are compared with the results obtained in [21] and [14]. Finally, an application of proposed PBVNS algorithm on a real world case of the shift design problem at TNT Express is presented and some managerial implications are stated.

5.1 Experimental Setup

5.1.1 Performance measures

Before going into the details of the experiments, we first need to define the performance measures that are going to be used to compare the results of our proposed algorithm with the ones obtained in [21] and [14]. One common approach is to look at the quality of the solutions and the computation time needed to reach these solutions. However, by only looking at these two criteria might not provide a full picture of the quality of the solution method and its suitability to other problem instances [23]. In [23], the authors propose seven criteria for qualitative comparison of nurse rostering approaches. Although these criteria are presented specifically for nurse rostering problems, we think that they can somewhat be generalized to other problems as well. A short overview of these seven criteria will be given now.

1. Expressive power. This refers to the ability of the model to represent a wide variety of real-world constraints and characteristics.
2. Flexibility. The ability to add/remove constraints at any time and the ability to effectively handle changing circumstances in the scheduling environment.
3. Algorithmic power. The main factors that determine the algorithmic power are efficiency and effectiveness. Effectiveness refers to the quality of the solutions and efficiency refers to the time it takes to generate a solution.
4. Learning. This refers to the capability of a system to improve its performance by gaining new knowledge/experience over time.
5. Maintenance. The ability of the system to update knowledge of the problem e.g. detect redundant or conflicting constraints and cases.
6. Rescheduling. This refers to the ability of a system to react to unforeseen disturbances in the real-world environment.
7. Parameter tuning. In order for algorithms to be able to generate good quality solutions, often the parameters that are employed in the algorithm need to be tuned carefully.

To be able to make use of all these seven criteria, one will need to have access to the implementation of the different algorithms that need to be compared. Unfortunately, we didn't have access to the implementations of [21] and [14]. Thus we can only compare our proposed algorithm with theirs using the common algorithmic power criteria as mentioned before. In other words, we'll compare the quality of the solutions generated by our algorithm with theirs. Although we don't make use of the other criteria from [23], it's still important to have mentioned them. This will create an awareness in the research community on how shift scheduling algorithms should be compared with each other.

5.1.2 ROTA test examples

To be able to run experiments, we will need data. For our experiments we used the test examples that were also used in [21] and [14]. These test examples were created by the research project Rota, which is undertaken by the

Database and Artificial Intelligence Group at the Vienna University of Technology in cooperation with Ximes Corp, and can be used as benchmark instances. These examples were randomly generated and they vary in their complexity. In each test example, the workforce requirements for a planning horizon of a week are specified in man hours needed for each time slot. The best known solution for each problem was first presented in [21]. In [14], the authors were able to find even better solutions to some of the ROTA examples as compared to [21]. The authors in both [21] and [14] encourage the use of the ROTA examples as benchmark cases for MSD algorithms. Therefore, we make use of these examples in order to be able to compare our results with theirs. All the test examples with their description are available in text files from *http://www.dbai.tuwien.ac.at/proj/Rota/benchmarks.html.* For a description on how these instances were created using a random instance generator refer to [21]. All the instances include the best solution and the objective value for this solution.

It is worth noting that each of these ROTA instances have a certain time interval: 15 minutes, 30 minutes or 60 minutes. We only used the instances with time interval of 60 minutes. The reason for this is that we integrated our algorithm in the manpower planning system of TNT Express and this system only has time intervals of 60 minutes. It was not straightforward to update the system to handle 15 or 30 minutes intervals and it was also outside the scope of this thesis to do that. Therefore, instead of using all 30 cases from [21] and [14], we'll use 10 of them.

5.1.3 Experiments and parameters

This section presents the set of experiments that we conducted for each ROTA test example during our computations. After some initial tests, we concluded that by conducting each experiment twenty times gave representative results. The objective values for each test example were calculated by taking the mean objective value over the twenty runs. The 95% confidence intervals belonging to the mean objective values were also calculated. A 95% confidence level means that 95% of this interval should include the population parameter that is being calculated [2]. An example on how to interpret the mean values and their confidence intervals will be given now. Consider a mean coverage percentage of 92% and a margin of error of 0.40 when calculating the 95% confidence interval. The 95% confidence interval for this mean is 92% $\pm$ 0.40. This means that 95% of the interval between 92-0.40=91.60% and 92+0.40=92.40% should include the population parameter that we are calculating. The stopping criteria for the PBVNS were set as follows. The number of iterations for PBVNS was set to 1000 and the number of iterations without improvement was set to 300. Higher values for the stopping criteria did not have an impact on the quality of the solutions. We'll now discuss each experiment in details and also the reason why it was conducted.

Population size

In the beginning of this Chapter, we showed that the PBVNS performs better than the VNS based method. Here, the population size was arbitrarily set to 10. Using more solutions at the same time during the search, can improve the performance of the algorithm. The reason for this is that during each step, more parts of the search space can be explored. This next experiment we conducted was to determine the best number of solutions to use during the optimization. Initially, we started out with 10 solutions. Then, we tried to use 20, 30, 50 and 100 solutions.

Reduced Variable Neighborhood Search

The following experiment was used to see the effect of improving the initial random solutions using RVNS. We want to have a good starting point for the search algorithm. By good, we mean that the starting point has to be very diverse. Using diverse initial solutions helps the algorithm in exploring more parts of the search space. Initially, our choice fell on a randomly created initial solution. In the literature however, the GVNS makes use of the RVNS to improve the initial random solution first, before going into the optimization step. When comparing the results of our proposed algorithm with a random initial solution and a random initial solution that has been improved by RVNS, we can see that the initial solution improved by RVNS has shown to give better results. However, it's worth stating that the number of iterations for RVNS is crucial in getting the right starting point for the optimization phase. When using too many iterations, the starting point will already be a somewhat optimized point. Thus it will be hard for the local search to escape from these areas during the search. For the computational results, we varied the number of iterations for RVNS with 50, 100 and 200 iterations.

Number of initial shifts and employees

In our proposed algorithm, we make use of a random number of initial shifts and a random number of employees. We experimented with different values to find out which value would give the best results. The number of initial shifts was varied between 2, 3, and 5 for each shift type. The number of initial employees per shift was varied using random numbers between 0 and 1, 3, 5, or 10. The reason for choosing these small values is as follows. For all four shift types, the number of initial shifts will be created for each of these shift types. Thus setting the number of initial shifts to a large number would mean a lot of shifts and a lot of employees. This is not desirable. Also, keep in mind that a shift appears on each day in the planning horizon of a week (7 days) and that each shift has a duration of several hours. Thus adding one employee to a shift of e.g. 8 hours would mean adding a total of (7x8=)56 man hours to the total number of man hours in a solution. If we use larger numbers here, then the initial solutions will contain too many employees and the PBVNS would need more iterations to be able to solve this problem.

Shake phase in VNS

The last experiment we did, concerned the shake phase in the VNS. In the literature, it is customary to use the shake phase to create a neighbor close to the current best solution and then start the local search from there. This may result in a better solution than the current one. The local search will run until no more improvement can be found and thus the algorithm has reached a local optimum. From here, the shake phase is repeated again to start the next local search using the neighbors of the current best solution.

The question that arises here is that why should only one shake be used? There is no guarantee that using one shake in the VNS will create the best candidate solution for the local search. It is possible that the optimal solution is in the neighborhood of the second or third neighbor of the current best solution. If we only use one shake in the VNS, then the optimal solution will not be found. Since all the neighbors of the current best solution are of less quality, the local search will not stay at that point. A solution to this problem would be to use multiple shakes instead of a single shake step. We tried different number of shakes in our search algorithm varying from 1, 2, and 3.

5.2 VNS vs PBVNS

In Chapter 4, we stated that early experimental results have shown that the VNS algorithm is not good enough to be used as our main algorithm. Therefore we extended the VNS based algorithm with elements from EAs. In this section, we'll make a comparison between the results of the VNS and PBVNS on the ROTA test examples and show that the PBVNS is indeed the better algorithm out of the two. The first comparison is done based on the total percentage of workload in the planning horizon that can be covered by the solutions generated by each algorithm. The population size of the PBVNS was arbitrarily set to 10. Each test case is solved twenty times in order to get a representative mean percentage of the total workload that was covered in the solutions. The corresponding confidence intervals were also calculated. Table 5.1 shows the results for both algorithms.

In order to verify that the results obtained by the PBVNS are significantly better than the ones from the VNS, we performed a statistical one tailed t-test for comparison of means. The t-test was performed with a significance level of 0.05 (or a confidence level of 95%). For more details on statistical t-tests, we advice the reader to refer to statistical literature i.e. [2]. Looking at the p-values in Table 5.1, we can conclude that for 6 of the 10 ROTA test exmaples, the PBVNS significantly gives better results than the VNS. For these cases the p-value is lower than the significance level of 0.05. For 3 other cases the p-values are respectively 0.091, 0.076 and 0.089, which are slightly higher than the significance level of 0.05. Thus the PBVNS almost generates significant better results for these three cases as well. This would have been the case if the sample size was larger or if the confidence level was a lower value.

		VNS	PBVNS	
ID	Workload	Mean Coverage %	Mean Coverage %	p-value
1	1099	95.85 ± 0.457	96.24 ± 0.320	0.091
3	1552	93.99 ± 0.487	95.25 ± 0.510	0.001*
5	1091	96.76 ± 0.404	98.09 ± 0.365	0.000*
6	585	96.44 ± 0.380	96.79 ± 0.362	0.076
15	231	97.08 ± 0.793	97.79 ± 0.704	0.089
18	1832	94.87 ± 0.261	95.98 ± 0.287	0.000*
20	1640	96.44 ± 0.362	96.66 ± 0.314	0.178
24	1065	96.28 ± 0.555	97.49 ± 0.310	0.000*
26	1598	94.28 ± 0.573	96.48 ± 0.395	0.000*
27	1725	94.80 ± 0.486	95.70 ± 0.352	0.007*

Table 5.1: Comparison between the results for VNS and PBVNS on the ROTA test examples. The mean coverage percentages and their 95% confidence intervals are shown for each case. The last column shows the one tailed p-value of the statistical t-test for comparison of the means with a significance level of 0.05. * indicates a significant difference.

At last, we also compared the results for both VNS and PBVNS by looking at the total number of workload covered by the generated solutions instead of the total percentages of workload covered by the solutions. Table 5.2 shows these results along with a one tailed t-test for comparison of means with significance level of 0.05. In this case, the PBVNS generates significantly better results than the VNS for 8 of the 10 ROTA test examples. Along with the previous results, we can conclude that the PBVNS is significantly a better algorithm for solving shift design problems than the VNS algorithm.

5.3 Computational results

This section presents the computational results of the PBVNS algorithm on the ROTA test examples. The computations were done on a Microsoft Windows XP Service Pack 3 computer with a 1.83GHz Intel CPU and 1.99GB of RAM. When evaluating the solutions, we looked at the percentage of the total workload that can be covered in the solutions found by the PBVNS. The following sections will report the results for each experiment that was discussed in the experimental setup. At last, we combined the best parameter values from each experiment and solved all the ROTA test examples one more time. These results are also presented in this section.

5.3.1 Population size

For the population size experiment, we varied the population size P with the values 10, 20, 50, and 100. The results for this experiment are shown in Table 5.3. The expectation was that by using more solutions during each iteration

		VNS	PBVNS	
ID	Workload	Mean Coverage	Mean Coverage	p-value
1	1099	1053 ± 6.43	1061 ± 3.80	0.003*
3	1552	1458 ± 8.10	1478 ± 7.60	0.000*
5	1091	1059 ± 3.29	1067 ± 3.47	0.000*
6	585	564 ± 3.37	564 ± 3.34	0.407
15	231	220 ± 1.99	226 ± 1.50	0.000*
18	1832	1741 ± 5.20	1755 ± 5.29	0.004*
20	1640	1584 ± 8.43	1588 ± 5.78	0.250
24	1065	1034 ± 4.29	1039 ± 3.43	0.029*
26	1598	1518 ± 6.46	1536 ± 5.52	0.000*
27	1725	1632 ± 10.72	1650 ± 6.95	0.010*

Table 5.2: Comparison between the results for VNS and PBVNS on the ROTA test examples. The mean total coverage and their 95% confidence intervals are shown for each case. The last column shows the one tailed p-value of the statistical t-test for comparison of the means with a significance level of 0.05. * indicates a significant difference.

could result in a better final solution. Looking at the values in Table 5.3, it can be seen that a population size of 50 gave the best results. For 6 out of the 10 cases, the best generated solutions were using this population size.

5.3.2 Reduced Variable Neighborhood Search

For the RVNS experiment, we varied the number of iterations for the RVNS. RVNS is used in the beginning of the PBVNS to improve the randomly generated solutions. The results for this experiment are shown in Table 5.4. It is quite clear to see that by using 50 iterations for the RVNS gave the best results. For 7 out of the 10 cases, this value gave the best results.

5.3.3 Number of initial shifts and employees

For the experiment about the number of initial shifts and the number of initial employees, the results are presented in Table 5.5 and Table 5.6. The values in Table 5.5 indicate that the best values for the number of initial shifts is 3. Table 5.6 shows that the best number of employees for the initial solutions should be between 0 and 3.

5.3.4 Shake phase in VNS

For the number of shakes experiment, we varied the value for the number of shakes in VNS between 1, 2, and 3. The results for this experiment are shown in Table 5.7. The values in Table 5.7 indicate that the best solutions are found using a shake number of 1. It is clear to see that increasing the number of shakes does not have a positive effect on the quality of the solutions. Using more shakes

	P=10	P=20	P=50	P=100
ID	Mean %	Mean %	Mean %	Mean %
1	96.24% ± 0.32	96.31% ± 0.37	96.61% ± 0.24*	96.27% ± 0.08
3	95.25% ± 0.51	95.45% ± 0.50	96.13% ± 0.21*	95.70% ± 0.71
5	98.09% ± 0.36	97.75% ± 0.42*	98.49% ± 0.17	97.67% ± 0.71
6	96.79% ± 0.36	96.66% ± 0.46	97.30% ± 0.25*	96.51% ± 0.33
15	97.79% ± 0.70	97.23% ± 0.87	98.44% ± 0.23*	97.92% ± 1.08
18	95.98% ± 0.28*	95.87% ± 0.36	95.84% ± 0.31	95.49% ± 0.23
20	96.65% ± 0.31	96.96% ± 0.40	97.21% ± 0.15*	97.00% ± 0.31
24	97.49% ± 0.31	97.39% ± 0.44	98.01% ± 0.33	98.08% ± 0.29*
26	96.48% ± 0.39	96.25% ± 0.49	96.72% ± 0.26*	95.52% ± 0.64
27	95.69% ± 0.35	95.67% ± 0.43	96.24% ± 0.44	96.41% ± 0.23*

Table 5.3: Overview of the results for population size experiment shown in percentages. The first column gives the ROTA example ID and the other columns show the mean coverage percentages along with their 95% confidence intervals. * indicates the best mean percentage in that row

	Iterations=50	Iterations=100	Iterations=200
ID	Mean %	Mean %	Mean %
1	96.24% ± 0.32*	96.14% ± 0.41	96.16% ± 0.31
3	95.25% ± 0.51	95.54% ± 0.49*	95.50% ± 0.69
5	98.09% ± 0.36*	97.89% ± 0.36	97.70% ± 0.62
6	96.79% ± 0.36*	96.63% ± 0.42	96.39% ± 0.30
15	97.79% ± 0.70*	97.25% ± 0.86	97.43% ± 0.59
18	95.98% ± 0.28*	95.61% ± 0.32	95.51% ± 0.20
20	96.65% ± 0.31	96.91% ± 0.38*	96.69% ± 0.46
24	97.49% ± 0.31*	97.02% ± 0.47	97.36% ± 0.32
26	96.48% ± 0.39	95.92% ± 0.42*	95.64% ± 0.65
27	95.69% ± 0.35*	95.34% ± 0.39	95.52% ± 0.36

Table 5.4: Overview of the results for number of iterations for RVNS experiment shown in percentages. The first column gives the ROTA example ID and the other columns show the mean coverage percentages along with their 95% confidence intervals. * indicates the best mean percentage in that row.

	NrShifts=2	NrShifts=3	NrShifts=5
ID	Mean %	Mean %	Mean %
1	93.04% ± 0.74	96.24% ± 0.32	97.03% ± 0.23*
3	94.96% ± 0.78	95.25% ± 0.51*	95.20% ± 0.40
5	97.20% ± 0.46	98.09% ± 0.36*	97.38% ± 0.28
6	96.41% ± 0.61	96.79% ± 0.36*	95.14% ± 0.77
15	97.66% ± 0.77	97.79% ± 0.70*	95.24% ± 1.03
18	95.31% ± 0.15	95.98% ± 0.28	96.51% ± 0.11*
20	95.87% ± 0.42	96.65% ± 0.31*	96.44% ± 0.47
24	95.87% ± 0.51	97.49% ± 0.31*	96.97% ± 0.66
26	95.01% ± 0.31	96.48% ± 0.39	97.78% ± 0.23*
27	95.55% ± 0.33	95.69% ± 0.35	96.10% ± 0.32*

Table 5.5: Overview of the results for the number of initial shifts experiment shown in percentages. The first column gives the ROTA example ID and the other columns show the mean coverage percentages along with their 95% confidence intervals. * indicates the best mean percentage in that row

	NrEmpl=3	NrEmpl=5	NrEmpl=10
ID	Mean %	Mean %	Mean %
1	96.24% ± 0.32*	96.05% ± 0.44	95.59% ± 0.33
3	95.25% ± 0.51	95.82% ± 0.23*	92.38% ± 0.97
5	98.09% ± 0.36*	97.36% ± 0.36	95.03% ± 0.55
6	96.79% ± 0.36*	96.64% ± 0.26	96.47% ± 0.32
15	97.79% ± 0.70*	96.71% ± 0.84	95.18% ± 0.91
18	95.98% ± 0.28*	95.62% ± 0.33	94.80% ± 0.48
20	96.65% ± 0.31*	96.21% ± 0.33	96.19% ± 0.60
24	97.49% ± 0.31*	97.22% ± 0.43	95.36% ± 0.34
26	96.48% ± 0.39*	96.29% ± 0.33	95.34% ± 0.59
27	95.69% ± 0.35*	95.40% ± 0.27	95.68% ± 0.54

Table 5.6: Overview of the results for the number of initial employees experiment shown in percentages. The first column gives the ROTA example ID and the other columns show the mean coverage percentages along with their 95% confidence intervals. * indicates the best mean percentage in that row

	NrShakes=1	NrShakes=2	NrShakes=3
ID	Mean %	Mean %	Mean %
1	96.24% ± 0.32*	95.53% ± 0.31	95.43% ± 0.37
3	95.25% ± 0.51*	94.74% ± 0.55	94.46% ± 0.46
5	98.09% ± 0.36*	97.08% ± 0.24	96.55% ± 0.49
6	96.79% ± 0.36*	95.85% ± 0.49	95.52% ± 0.50
15	97.79% ± 0.70*	96.45% ± 1.39	94.41% ± 1.01
18	95.98% ± 0.28*	95.09% ± 0.68	95.49% ± 0.29
20	96.65% ± 0.31*	96.90% ± 0.20	95.90% ± 0.33
24	97.49% ± 0.31*	95.91% ± 0.46	96.17% ± 0.53
26	96.48% ± 0.39*	95.19% ± 0.77	94.66% ± 0.48
27	95.69% ± 0.35*	95.30% ± 0.37	95.61% ± 0.33

Table 5.7: Overview of the results for the number of shakes in VNS experiment shown in percentages. The first column gives the ROTA example ID and the other columns show the mean coverage percentages along with their 95% confidence intervals. * indicates the best mean percentage in that row

seems to move the algorithm away from the current good neighborhood that the search is in, resulting in solutions of less good quality.

5.3.5 Best results

After experimenting with the different parameters, we obtained the best set of parameter values for the PBVNS:

- the population size is 50;
- the number of iterations for RVNS is 50;
- the number of initial shifts is 3;
- the number of initial employees is between 0 and 3;
- the number of shakes for VNS is 1;

Using this combined set of parameters, we solved the ROTA test examples one more time. During each run, we solved all the 10 test examples one after the other. This sequence was repeated twenty times. The mean computation time for each complete run was around 60 seconds. Thus the mean computation time to solve one case was around 6 seconds. Between the cases themselves, there was big noticeable difference in computation time. Only the cases containing the higher total of workload took a bit longer to solve than the ones containing less workload. When evaluating the solutions, we focused on the percentage of the total workload that can be covered in the solution found by the PBVNS. Additionally, we looked at the mean percentage of overcover and undercover of our solutions and also the number of shifts used for each test example. Table 5.8 shows the results of our algorithm on the ROTA test examples.

		Coverage	Undercover	Coverage
ID	Workload	Mean %	Mean %	Mean %
1	1099	96.24% ± 0.32	3.76% ± 0.31	3.58% ± 0.70
3	1552	95.25% ± 0.51	4.75% ± 0.51	3.61% ± 0.31
5	1091	98.09% ± 0.36	1.91% ± 0.71	1.34% ± 0.63
6	585	96.79% ± 0.36	3.21% ± 0.40	5.47% ± 0.92
15	231	97.79% ± 0.70	2.21% ± 0.31	2.82% ± 0.37
18	1832	95.98% ± 0.28	4.02% ± 0.39	4.13% ± 0.34
20	1640	96.65% ± 0.31	3.35% ± 0.31	2.98% ± 0.37
24	1065	97.49% ± 0.31	2.51% ± 0.50	3.01% ± 0.40
26	1598	96.48% ± 0.39	3.52% ± 0.46	3.50% ± 0.25
27	1725	95.69% ± 0.35	4.31% ± 0.46	4.20% ± 0.26
avg	-	96.65% ± 0.57	3.35% ± 0.57	3.46% ± 0.67

Table 5.8: Overview of the results for the test cases using the best set of parameters. The first column gives the ROTA example ID. The second column indicates the total workload over the planning horizon of a week given in man hours. The mean percentage of workload that has been covered by our solutions is shown in the third column along with its 95% confidence interval. Finally, the last two columns show the percentages of undercover and overcover for the planning horizon with their 95% confidence interval.

For the smallest test case (ROTA ID=15), our algorithm was able to find the optimal solution in some of the runs. For all the cases (of different sizes), the best solution found by our algorithm covered a minimum of 95% of the total workload over the entire planning horizon. The maximum undercover was 4.75% and the maximum overcover is 5.47% for all the test cases. The confidence intervals for all the mean values are smaller than 1. Thus we can conclude that solutions found by the PBVNS are quite stable. The mean percentage of the total workload covered for the test examples is 96.65%, with a mean undercover of 3.35% and a mean overcover of 3.46%. From these results, we can conclude that our proposed algorithm is able to generate good quality solutions for the ROTA test examples. Even though the 100% coverage solutions are not generated, the generated solutions are good enough to be used to create a tactical schedule.

5.3.6 Comparison with another method from the literature

Table 5.9 shows the results of our algorithm in comparison to the results obtained by Musliu et al in [21] and Di Gaspero et al in [14]. For these ten ROTA test examples, the results from Musliu and Di Gaspero were identical. Thus we put them together in the same column in Table 5.9 and we'll refer to these results as the ones from Di Gaspero. In Table 5.9, the total workload and

		Di Gaspero			PBVNS		
ID	Work	Cover	U/O	NrShifts	Mean Cover	U/O	NrShifts
1	1099	1099*	0/0	8	1095 ± 3.80	4/38	8
3	1552	1552*	0/0	11	1520 ± 7.61	32/25	8
5	1091	1091*	0/0	8	1087 ± 3.47	4/0	9
6	585	585*	0/0	7	579 ± 3.34	6/32	8
15	231	231*	0/0	3	231 ± 1.50*	0/0	3
18	1832	1832*	0/0	13	1794 ± 5.29	38/39	10
20	1640	1640*	0/0	9	1614 ± 5.78	26/16	9
24	1065	1065*	0/0	9	1061 ± 3.43	4/0	9
26	1598	1598*	0/0	11	1590 ± 5.52	8/8	9
27	1725	1725*	0/0	9	1683 ± 6.95	42/38	10

Table 5.9: Comparison of the results found by the PBVNS algorithm and the ones found by Di Gaspero. The first column shows the ID of the test example. The second column shows the total workload for that planning horizon. Then the mean coverage (Cover), mean undercover (U), mean overcover (O) and number of shifts (NrShifts) for both solution methods are shown together with their 95% confidence interval. * indicates the best mean coverage for that row.

the mean coverage by our algorithm and the coverage by the algorithm of Di Gaspero are given in man hours. The algorithm by Di Gaspero is able to generate the optimal solution for all the test cases. This can be seen in Table 5.9 by the fact that all the workload is covered in their solutions and that there is no undercover and or overcover either. Our proposed PBVNS was only able to generate the optimal solution for one of the test cases. For all the other cases, we only found a near optimal solution. From Table 5.9, we can conclude that even though our algorithm generates good quality solutions covering an average of 95% of the total workload, the algorithm by Di Gaspero still outperforms our PBVNS in terms of quality of solutions.

5.4 Real world case

The goal of this thesis was to design an algorithm that can solve real world instances of shift design problems. Since the performance of our proposed algorithm on the test examples gave quite good results, we also want to evaluate the performance of our algorithm on a real world case. For our real world case, we looked at the shift design problem that TNT faces on the depot location at Arnhem in the Netherlands. TNT is an express service provider and offers a wide range of mail and express distribution services worldwide for businesses and consumers. Consignments consisting of parcels and freight are delivered on the daily basis from numerous locations to other locations through the entire supply chain. At first, all the consignments are picked up from the origin locations where the customers dropped them off and they are driven to the nearest depot location where they are unloaded, processed and handled. Afterwards, the consignments are loaded on another vehicle and driven to their destinations where they are delivered to the customers.

At each location in the supply chain of TNT, there is a lot of work that needs to be done by the employees. A few examples of activities that needs to be done at these locations are unloading of vehicles, sorting consignments, processing consignments, handling consignments, loading of vehicles, etc. TNT now has a lot of people working at these locations in order to get all the work done in time. It is often the case that the depot manager doesn't know exactly how many people he has to schedule for the next planning horizon in order to get all the work done. He creates the schedule based on seasonal trends, rule of thumb or his own expert knowledge. This can easily result in inefficiencies when too many people are scheduled or it can result in capacity issues when not enough people are scheduled. What TNT would like, is to be able to create a tactical planning for the next planning horizon that minimizes inefficiencies and capacity issues while getting all the work done in time.

For the real world application of our algorithm, we tried to create a tactical planning for the depot location at Arnhem in the Netherlands. A picture taken from the inside of this location is shown in Figure 5.1. The data used for this

Figure 5.1: A picture of the depot location at Arnhem in the Netherlands where TNT Express uses the manpower planning system to create tactical plans.

case consisted of the estimated workforce requirements for the planning horizon of a week at the Arnhem depot. A picture of these workforce requirements is shown in Figure 5.2. There are quite some differences in the characteristics of the Arnhem case compared to the ROTA test examples. The total workforce requirements over the entire planning horizon for the Arnhem case is 6784 man

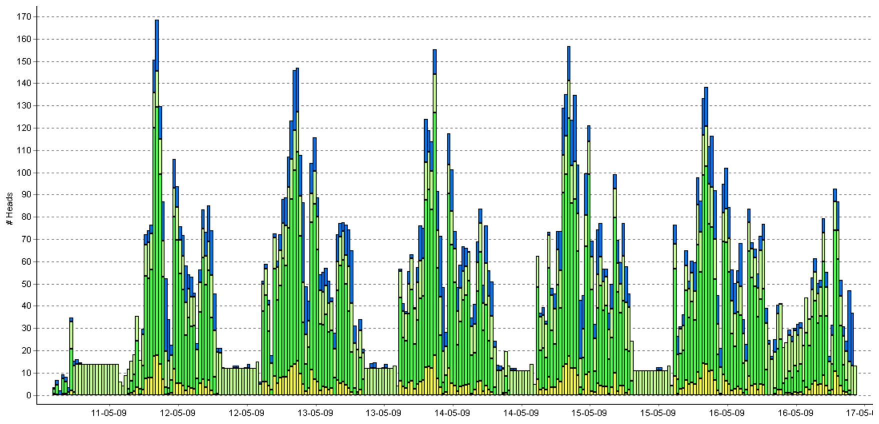

Figure 5.2: Workforce requirements of the TNT Express location at Arnhem in the Netherlands taken from the manpower planning system.

hours. This is almost four times as much as the highest total workload for the test cases. There the maximum workforce requirements was 1832 man hours. The number of timeslots for both cases are the same. However, in a lot of the test cases there are time slots with 0 workforce requirements, that is not the case in the Arnhem case. Finally, the highest peak in workforce requirements for the test examples was 31. For the Arnhem case this is 169, which is more than five times the highest peak for the test cases. With all those differences listed, the main question now is whether our algorithm is able to find a good quality solution and create a good tactical schedule for this problem.

5.4.1 Experimental setup

For the real world case, we used the same experimental setup as for the ROTA test examples. The same experiments were done here: population size, number of iterations for RVNS, number of initial shifts/employees and the number of shakes in VNS. The default values for these parameter are set equal to the best set of parameters for the test examples. When experimenting with one of the parameters, all the other parameters are kept at the default values. Each experiment was ran 20 times in order to get a representative mean objective value and its corresponding 95% confidence interval. There are three scenario's that were calculated during each run. The first scenario was done using the same weight factors that have been used during our experiments. In the last two scenario's, two higher penalty values were assigned to the undercover time periods. In the real world, managers wouldn't mind too much that there are too many people on the floor. But the case of not having enough people on the floor and thus not getting the work finished in time is often considered to be unacceptable. A summary of the weight factors for the three scenario's is as

	P=10	P=20	P=50	P=100
Scene	Mean %	Mean %	Mean %	Mean %
1	76.56% ± 0.46	76.74% ± 0.52	76.92% ± 0.49	77.61% ± 0.34*
2	91.95% ± 0.70	92.79% ± 0.65	93.55% ± 0.38	93.81% ± 0.33*
3	95.75% ± 1.29	95.86% ± 0.94	97.53% ± 0.45*	97.06% ± 0.54

Table 5.10: Overview of the results for population size experiment on the real world case shown in percentages. The first column gives the scenario ID and the other columns show the mean coverage percentages and their 95% confidence intervals. * indicates the best mean percentage in that row

	Iterations=50	Iterations=100	Iterations=200
Scene	Mean %	Mean %	Mean %
1	76.70% ± 0.68*	75.38% ± 0.52	75.76% ± 0.55
2	97.38% ± 0.53*	92.21% ± 0.57	92.83% ± 0.60
3	97.38% ± 0.53*	96.10% ± 0.81	96.41% ± 0.97

Table 5.11: Overview of the results for number of iterations for RVNS experiment on the real world case shown in percentages. The first column gives the scenario ID and the other columns show the mean coverage percentages and their 95% confidence intervals. * indicates the best mean percentage in that row.

follows:

- Scenario 1: (w_1=1, w_2=1 and w_3=60);
- Scenario 2: (w_1=5, w_2=1 and w_3=60);
- Scenario 3: (w_1=10, w_2=1 and w_3=60);

The computational results of the three scenario's for the real world case will be presented now.

5.4.2 Computational results

Population size

Table 5.10 shows a summary of the results for the population size experiment for the real world case. The population size that gave the best results for this test case was 100.

Reduced Variable Neighborhood Search

The results for the experiment concerning the number of iterations for RVNS are summarized in Table 5.11. The best results were obtained by setting the number of iterations for RVNS to 50.

	NrShifts=2	NrShifts=3	NrShifts=5
Scene	Mean %	Mean %	Mean %
1	71.98% ± 0.32	75.42% ± 0.96	77.53% ± 0.67*
2	91.97% ± 0.65	93.30% ± 0.71	94.12% ± 0.31*
3	94.56% ± 0.41	97.97% ± 0.11	98.64% ± 0.27*

Table 5.12: Overview of the results for the number of initial shifts experiment on the real world case shown in percentages. The first column gives the scenario ID and the other columns show the mean coverage percentages and their 95% confidence intervals. * indicates the best mean percentage in that row.

	NrEmpl=3	NrEmpl=5	NrEmpl=10
Scene	Mean %	Mean %	Mean %
1	75.42% ± 0.96	76.53% ± 0.45	78.12% ± 0.23*
2	93.30% ± 0.71	93.42% ± 0.78	94.04% ± 0.54*
3	96.97% ± 0.11	97.34% ± 0.36	98.01% ± 0.23*

Table 5.13: Overview of the results for the number of initial employees experiment on the real world case shown in percentages. The first column gives the scenario ID and the other columns show the mean coverage, undercover and overcover percentages along with their 95% confidence intervals. * indicates the best mean percentage in that row

Number of initial shifts and employees

Tables 5.12 and 5.13 show the results for the experiments regarding the number of initial shifts and the number of initial employees that should be used when solving the real world case. Using 5 initial shifts and 10 initial employees gave the best quality solutions.

Shake phase in VNS

For the number of shakes experiment, we obtained the results that are presented in Table 5.14. The values indicate that the best solutions are found using a shake number of 1. Using a higher number did not have a positive effect on the quality of the solutions.

5.4.3 Best results real world case

After having experimented with the different parameter values, we found the best set of parameters for this real world case:

- the population size is 100;
- the number of iterations for RVNS is 50;
- the number of initial shifts is 5;

	NrShakes=1	NrShakes=2	NrShakes=3
Scene	Mean %	Mean %	Mean %
1	75.42% ± 0.96*	75.36% ± 0.50	73.03% ± 0.51
2	93.30% ± 0.71*	92.85% ± 0.42	92.07% ± 0.65
3	97.97% ± 0.11*	95.74% ± 0.65	95.57% ± 0.97

Table 5.14: Overview of the results for the number of shakes in VNS experiment on the real world case shown in percentages. The first column gives the scenario ID and the other columns show the mean coverage percentages and their 95% confidence intervals. * indicates the best mean percentage in that row

		Coverage	Undercover	Overcover
Scene	Workload	Mean %	Mean %	Mean %
1	6784	85.19% ± 0.92	14.81% ± 1.97	10.54% ± 1.47
2	6784	95.67% ± 0.25	4.33% ± 0.93	28.79% ± 2.03
3	6784	99.85% ± 0.13	0.15% ± 0.10	39.23% ± 4.59

Table 5.15: The best computational results of three scenario's conducted on the real world case data from TNT Express. The first experiment is based on the default weight factors. Experiments two and three have a higher penalty for undercover time periods.

- the number of initial employees is between 0 and 10;
- the number of shakes for VNS is 1;

We used this combined set of parameters to solve the real world case again. The results for the three real world scenario's are summarized in Table 5.15. The mean computation time to solve the real world case was around 20 seconds. There was no substantial difference in computation time for the different scenario's. For the normal weight factors, our algorithm is able to generate a solution that can cover a mean of 85.19% of the total workload with an undercover of 14.81% and an overcover of 10.54%. This result is quite good considering the fact that this real world case is four times the size of the test examples and the highest peak of the workload is also much higher. We can still cover around 85% of the total workload. In the next scenario's, a higher penalty was given to the undercover time periods. It's clear to see in 5.15, that changing the ratio of the weights can have a significant impact on the quality of the solutions generated by our algorithm. In the first scenario, our algorithm was only able to find a solution that covered a mean 85.19% of the total workload. By changing w_1=1 to w_1=5, our algorithm found a solution that covered a mean 95.67% of the total workload. However, there is also a downside to this. Because the weight for undercover is now higher than the one for overcover, a lot more overcover will be present in the best solutions. Since the algorithm is more focused on reducing the total undercover. This effect is even more clear when we look at the results of the third scenario. Here, our algorithm found a solution that covers a mean 99.85% of the total workload. However, this solution contains almost 40% of overcover over the entire planning horizon. This number is too high for this solution to be considered a good quality solution. It should be clear now that by experimenting with the weight factors, it's possible to find the solution that best matches the needs of the person trying to solve the shift design problem.

5.5 Managerial implications

From our computational results, we can conclude that depot and hub managers can make use of our proposed algorithm to create a workforce schedule at a tactical level. The number of employees specified and the shift times in this schedule should give a good indication of the next planning horizon for when people are needed and also how many of them are needed. This tactical schedule will help the managers in planning the employees they need at their location.

However, the managers should keep in mind that the created schedules are at a tactical level. On the operational level, the manager will still have to decide who is going to work the shifts that are specified in the solution. Also, there can be some last minute factors that have an impact on the feasibility of the schedule during operations. It can be the case that due to seasonality the workforce requirements are much more than initially expected. For these cases, the manager will have to rely on his expert knowledge to deal with the new situation. Or he can model the new situation in the manpower planning system and run our proposed algorithm again to create another tactical schedule for this new situation.

Chapter 6

Conclusion and future research

The *min*-Shift Design problem is an important shift scheduling problem that appears in many different industries. For the postal company TNT Express, there was a need to develop an algorithm that can be used to solve shift design problems. This algorithm would be integrated in their manpower planning system, making it possible for them to automatically create tactical workforce schedules for the coming planning horizon. In this thesis, we studied the MSD problem and we developed an algorithm to solve it. The implemented algorithm is easy to use, has four parameters that can be understood without too much technical knowledge and the computation effort needed to generate solutions is not too high.

The initial algorithm we designed, was fully based on the meta-heuristic VNS. Some preliminary experiments showed that this algorithm needed improvements. So we extended the basic framework of the VNS search algorithm by transforming it into the so called Population Based Variable Neighborhood Search. Instead of using only one configuration at each iteration during the search, several configurations are used simultaneously now. At each iteration, new solutions are created and the population is updated. For updating the population of solutions, we took inspiration from the EAs and made use of the steady state replacement method.

In the experimental part, we evaluated different components of our proposed search algorithm. We discussed in detail how the parameters were determined. Then, we compared the performance of our algorithm to two other approaches in the literature using the ROTA benchmark examples. Our computational results show that the PBVNS is able to generate good quality solutions in a short amount of computation time and without having to set too many parameters. For some of the cases, it was even able to generate the optimal solution. Still our algorithm is slightly inferior to the two other approaches in the literature. As they are able to generate the optimal solution for all the ROTA test examples.

Finally, we applied the PBVNS on a real world case from the postal industry using data from TNT Express. The total workload in this case was four times more than the ones in the ROTA examples. And although the optimal solution was not found for this case, the solution generated by the PBVNS was of good quality. More than 90% of the total workload over the entire week was covered in the best solution. By adjusting the weights for the different components in the objective function, the PBVNS was able to generate a solution that covered all the workload for the entire week. Thus we can conclude that the PBVNS can be used to solve real world cases of the MSD problem.

A first recommendation for future research would be to improve our algorithm. The solutions generated by the PBVNS are of good quality, but it is not always able to generate the optimal solution. The main issue that needs to be tackled is the phase of choosing the right point to start the local search. In our algorithm, this is done randomly as described in the literature. However, this is not the most effective way to do it, as the probability of choosing the right point randomly is very small. It's possible that a different approach would be better. For example, using a type of acceptance function like in Simulated Annealing, including some kind of TABU mechanism or using a Memetic approach. In the Memetic approach, a GA is hybridized with a local search. The GA can be used for exploration of the search space and the VND can be used as a local search to improve each individuals fitness by hillclimbing.

The next recommendation for future research would be to analyze the weight factors in the MSD problem and find out what the best weights should be. Since the MSD problem is an optimization problem with several objectives, it is up to the user that is solving the problem to determine the weights. However, we have shown in this thesis that if these weights are not set correctly, then there is no guarantee that a good quality solution can be found. The goal would be to find the best set of weights that makes a good tradeoff between overcover, shortage and the number of shifts used. It's also possible to use another approach for the MSD problem that doesn't need the weight factors. One such approach would be to incorporate fuzzy criterion into the problem formulation. Three membership functions will need to be defined for the different objectives. The overall objective value of a solution would be the aggregation of the three membership functions using a fuzzy t-norm e.g. the minimum operator.

A last direction for future research would be to study some other forms and extensions of the MSD problem. Since this problem arises in so many different industries, it is likely that variations to this generic problem can be investigated. For instance, in the postal industry it is often the case that pieces of work are put aside into a so called buffer and then executed at a later moment in time. This results in better utilizations of the employees. By adding the possibility to move a piece of work to a later moment in time increases the complexity of the MSD problem substantially. Another possible extension for the MSD problem would be to take skill levels into account. Here, the solution to the MSD problem would not only contain shift times and the number of employees, but also the skill level needed by the employees working that shift.

Bibliography

[1] E.H.L. Aarts and J.K. Lenstra. *Local Search in Combinatorial Optimization.* Wiley-Interscience, Chichester, England, 1997.

[2] A.D. Aczel and J. Sounderpandian. *Complete Business Statistics.* McGraw-Hill, 2002.

[3] U. Aickelin and K. Dowsland. An indirect genetic algorithm for a nurse scheduling problem. *Computers and Operations Research*, 31(5):761–778, 2004.

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

[5] F. Bellanti, G. Carello, F. Della Croce, and R. Tadei. A greedy-based neighborhood search approach to a nurse rostering problem. *European Journal of Operational Research*, 153(1):28–40, 2004.

[6] M. Brusco and L. Jacobs. A simulated annealing approach to the solution of flexible labor scheduling problems. *Journal of the Operational Research Society*, 44(12):1191–1200, 1993.

[7] E.K. Burke, P. Cowling, P. De Causmaecker, and G. Vanden Berghe. A memetic approach to the nurse rostering problem. *Applied Intelligence*, 15(3):199–214, 2001.

[8] E.K. Burke, T. Curtois, G. Post, R. Qu, and B. Veltman. A hybrid heuristic ordering and variable neighborhood search for the nurse rostering problem. *European Journal of Operational Research*, 188(2):330–341, 2008.

[9] E.K. Burke, P. De Causmaecker, S. Petrovic, and G. Vanden Berghe. Variable neighborhood search for nurse rostering problems. *in Metaheuristics: Computer Decision-Making (edited by M.G.C. Resende and J.P. de Sousa)*, Chapter 7:153–172, 2003.

[10] E.K. Burke, P. De Causmaecker, G. Vanden Berghe, and H. Van Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7(6):441–499, 2004.

[11] E.K. Burke and G. Kendall. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Springer, 2005.

[12] B. Cheang, H. Li, A. Lim, and B. Rodrigues. Nurse rostering problems - a bibliographic survey. *European Journal of Operations Research*, 151:447–460, 2003.

[13] G. Dantzig. A comment on edie's traffic delay at toll booths. *Operations Research 2*, pages 339–341, 1954.

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

[15] T. Emden-Weinert and M. Proksch. Best practice simulated annealing for the airline crew scheduling problem. *Journal of Heuristics*, 5(4):419–436, 1999.

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

[17] F. Glover and G.A. Kochenberger. *Handbook of Metaheuristics*. Kluwer Academic Publishers, Boston, 2003.

[18] W J. Gutjahr and M.S. Rauner. An aco algorithm for a dynamic regional nurse-scheduling problem in austria. *Computers and Operations Research*, 34(3):642–666, 2007.

[19] P. Hansen and N. Mladenovi'c. Variable neighborhood search for the p-median. *Location Science*, 5(4):207–226, 1997.

[20] N. Musliu. Applying tabu search to the rotating workforce scheduling problem. *The 5th Metaheuristics International Conference (MIC'03)*, 2003.

[21] N. Musliu, A. Schaerf, and W. Slany. Local search for shift design. *European Journal of Operations Research*, 153:51–64, 2004.

[22] D. Parr and J.M. Thompson. Solving the multi-objective nurse scheduling problem with a weighted cost function. *Annals of Operations Research*, 155(1):279–288, 2007.

[23] S. Petrovic and G. Vanden Berghe. Comparison of algorithms for nurse rostering problems. *Working Paper*, University of Nottingham, UK, 2008.

[24] Y. Shen and R.S.K. Kwan. Tabu search for driver scheduling. *in: S. Voss, J.Daduna (Eds.), Computer-Aided Scheduling of Public Transport, Lecture Notes in Economics and Mathematical Systems*, 505:121–136, 2001.

[25] G. Thompson. Improved implicit modeling of the labor shift scheduling problem. *Management Science 41*, (4):595–607, 1995.

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