Erasmus University Rotterdam
Erasmus School of Economics

Master Thesis

A column generation approach to solving the multi-activity shift scheduling problem: a case study for Schiphol airport security

Olaf Legtenberg
533392

5th of November 2020

MSc Econometrics & Management Science
Track Operations Research & Quantitative Logistics

Supervisor:
Dr. R. Spliet
Co-reader:
Dr. W. van den Heuvel

Supervisors:
Dr. B. van Goudoever
J. Onderdijk MSc

The content of this thesis is the sole responsibility of the author and does not reflect the view of the supervisor, second assessor, Erasmus School of Economics or Erasmus University.

Abstract

In this thesis, an approach to solve the multi-activity shift scheduling problem (MASSP) with multiple shift types and no fixed starting points is presented. In the problem, security officers for customs need to be scheduled over the locations where officers are necessary. The schedule has to take in to account rules and regulations regarding the shift, the breaks as well as other restrictions. Currently, the problem is solved using a binary programming (BP) model. In this thesis, a column generation approach is designed for the problem. The column generation (CG) has both a heuristic and exact variant for the pricing problem. The exact pricing problem is modelled as a resource-constrained shortest path problem and is solved using a pulse algorithm. The heuristic method is based on a large neighbourhood search. In this thesis, two experiments are performed to determine what the performance of the methods is. The measurements for this are the average discount rate (VDC), the average weighted VDC, the standard deviation of the VDC and their solve time. The VDC is given by dividing the total number of hours planned by the number of hours demanded. The measurements show that the column generation methods have improved results for both the VDC and the solving time in comparison to the benchmark method. The results show that the CG Heuristic is the best working method. In the second experiment, the effect of increasing the number of filters in the MASSP is determined for the methods. Moreover, the effect of time on the quality of solutions of the methods is determined. We find that the CG heuristics VDC is less heavily impacted by increasing the number of filters and is also less dependent on the computation time.

Keywords: Multi-activity shift scheduling problem, column generation, resource-constrained shortest path problem, shift scheduling, case study, pruning strategies, binary programming

Contents

1 Introduction

Airport security is of growing importance since the terrorist attacks of 9/11. Quality of screening drastically increased since that time. This paper concerns security scheduling at Schiphol Airport in the Netherlands. Schiphol Airport is the twelfth largest airport in the world and the third-largest airport in Europe. The airport is located near Amsterdam, the Netherlands. The number of passengers of Schiphol has been drastically increasing over the last few years (CBS (2017)). In 2019, about 71 million passengers travelled through Schiphol Airport, which boils down to on average 194,000 passengers a day (CBS (2020)). Passengers departing to or arriving from non-Schengen countries need to be thoroughly searched to prevent safety hazards. These searches are carried out at security checkpoints scattered around the airport. These security checkpoints are called filters. The filters are divided into lanes. The filters are divided into three security zones and the handling of security of these zones is outsourced to private firms. Schiphol Airport pays these firms based on the number of lanes Schiphol Airport demands to be open to handle the passengers. For these firms, personnel costs are a considerable amount of their total operating cost. Therefore, inefficient scheduling of personnel has a great impact on the profitability of these firms. In this thesis, I propose an efficient approach to fulfil the demands of Schiphol Airport while minimising the amount of personnel necessary.

As particular filters have different types of flight, such as different destinations or a large number of transferring passengers, Schiphol Airport has irregular demand patterns for security agents. In addition, every security filter has an independent and dissimilar demand pattern. Consequently, in an efficient solution, security agents can relocate to a different filter to prevent unproductive working time.

Efficient scheduling of workers has been of growing relevance as the 24-hour economy has become more established. Moreover, due to the increase in computational power, opportunities to find efficient solutions have been increasing. These reasons give rise to the number of companies focusing on solutions for the efficient scheduling of employees. One of the companies developing solutions is Syntro, based in Amstelveen, the Netherlands. Syntro is a consultancy firm specialised in sustainable and efficient personnel scheduling, as well as work issues related to irregular working hours. Syntro has provided me with the opportunity to research efficient scheduling of security officers at Schiphol Airport for one of the private security firms that are responsible for the security of several passenger filters.

Personnel in this sector is often scheduled as a team. A team consists of four security officers and one team leader. Scheduling in teams has a few advantages. First of all, when officers stand at a filter, practically no social communication is possible between colleagues. As they are scheduled together for the whole day, they have the opportunity to interact during breaks. Secondly, during a shift, workers rotate between different jobs within a lane. In case there would be individual schedules, it would be unclear whose job would be taken over after someone's shift has ended. In

case the team is scheduled together, this would be clear. Thirdly, once the activities of a team end at a lane, there is a short period in which the new team needs to get installed. If this would happen every time someone's shift ends, there would be a more regular shutdown, which would also result in inefficiency. Finally, if one would consider individual scheduling, the gender of the security officers has to be taken into account. This would increase the number of constraints and hence the difficulty of the problem. In case team scheduling is considered, this is not of such large importance. To summarise, scheduling in teams results in less freedom to create rosters in comparison to individual scheduling. Therefore, the results are slightly less efficient. However, the social and practical advantages of team scheduling outweigh the inefficiency caused by team scheduling. Therefore, this thesis considers team scheduling.

Solutions for scheduling personnel already exist. The solution Syntro currently provides works as follows. First, all possible shifts are generated. A shift is a series of activities with a set start and end time. The set of possible shifts is then pruned in both a logical and a random way. In logical pruning, unlikely shift scenarios are pruned, while in random pruning the number of possible shifts is reduced. The remaining set of shifts is then formulated as a Mixed Integer Linear Programming model (MIP) and is solved using a commercial solver. The reasons a new approach has to be established are three-fold. First of all, the current approach is relatively unscalable. In case more filters are included, the number of possible shifts grows exponentially. Therefore, relatively more shifts should be pruned to obtain a feasible solution. Hence, attaining a solution (or having a small optimality gap) might be impossible if more filters are included. Secondly, the problem is currently formulated using a MIP. As the number of remaining shifts after pruning is substantial, so is the MIP. The commercial solver is not able to fully solve the problem to optimality within the time limit of ten minutes. Lastly, there are multiple job types which can be separated. For every individual type of job and every individual day, the commercial solver has been given a maximum time of ten minutes to solve the problem. Solving the problem for only one type thus takes up to 70 minutes for a week. This is considered too much. Therefore, the new approach needs to be faster in finding a solution. In this thesis, only one type of job is considered. The reason for this is, although the problems for other job types are relatively similar, some regulations are slightly different. Therefore, this thesis considers only one type of job.

The objective of this thesis is to create an efficient method for scheduling security teams to fulfil the demand, thereby minimising the number of hours necessary to create a feasible schedule. During their shift, officers can transfer between filters to increase the efficiency of the schedule. The performance of the method will be evaluated by using data that is available to the author. The data contains the demand for security officers at different filters. There are two information points. These points are interconnected. In the first information point, about four weeks before the schedule is used, the expected demand is provided. Shifts are created based on the demand information. A few days before the planning is in actual use, more accurate demand becomes available. The schedule can then be adjusted to this demand. Nonetheless, to remain fair to their employees as

well as keeping the health of workers in mind and to prevent additional payment, the collective labour agreement imposes restrictions on the company regarding the restructuring of shifts and more particularly, their start and end times. This is done in such a way that a sufficient number of shifts needs to come back with similar starting times and shifts lengths in the second process. To increase understanding of the process, a simplification of the process is included in Figure 1. The process of adjusting shifts to the new more accurate demand is called coupling. The coupling procedure makes sure that there is a sufficient percentage of shifts that return for the second demand point. Note that schedule in the figure refers to a set of shifts.

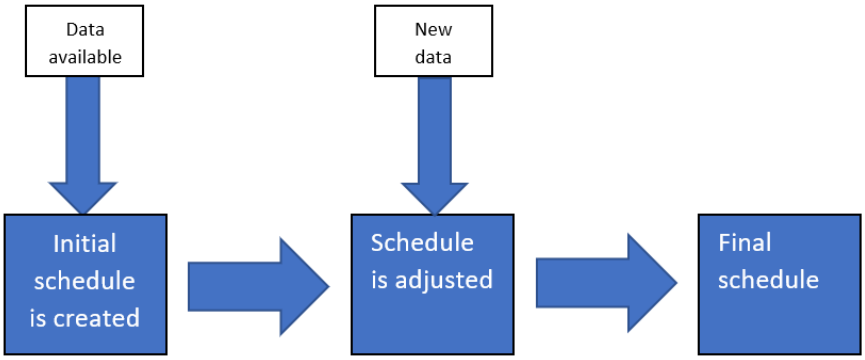

Figure 1: Process of creating a schedule

To be more precise, the main question that we want to answer in this research is the following: What is the effect of using a column generation approach, compared to the current solution, not only in terms of solution value but also in terms of computational efficiency as well as scalability?

The thesis is outlined as follows. In Section 2, the problem is described in a more detailed manner, while in Section 3 existing literature regarding multi-activity shift scheduling is discussed. In Section 4 I will elaborate on the methods applied in this thesis. In Section 5, the data used in this thesis is described. Subsequently, the experiments and their results are discussed in Section 6. In Section 7 conclusions are drawn from the experiments. Lastly, in Section 8, opportunities for future research are discussed.

2 Problem description

Schiphol Airport is a large airport that consists of multiple terminals. Every terminal has security filters. These filters are locations at which passengers can be searched and pass through security to go to their gate. The filters consist of lanes. A security team works at a lane. In this thesis, we are interested in the filter scheduling process. The lane assignment problem is a different problem and is hence not discussed in this thesis. The filter scheduling problem outlined in this thesis is described in the literature as the multi-activity shift scheduling problem (MASSP). In this thesis, we are interested in scheduling security teams in such a way that the demand for security teams

is satisfied while minimising the number of hours required. Moreover, shifts have to satisfy the rules and regulations as set in the collective labour agreement and the labour laws. To schedule the security teams efficiently, teams can be repositioned to a different filter during their shifts.

A shift i is defined as a sequence of activities, walking time and break time which has a start time s_i and an end time f_i and has a prespecified length. Moreover, let a working day be defined as a day with a shift with a starting and ending time defined.

Mathematically, let $\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$ and $\mathcal{W}$ be the set of shifts, the set of activities, the set of time intervals and the set of shift lengths, respectively. Let $z \in \mathcal{Z} = \{0, 1\}$ represent the two information moments, where information moment $z = 0$ is four weeks before and information moment $z = 1$ is the information moment in the week before the schedule is used. The information moment informs us about which set of demands is used in addition to which set of restrictions. Set of shifts $\mathcal{S}$ consists of subsets, dependent on the information point $z \in \mathcal{Z}$ and the shift type $w \in \mathcal{W}$. To be precise, $\mathcal{S} = \mathcal{S}^0 \cup \mathcal{S}^1$. $\mathcal{S}^z$, $z \in \mathcal{Z}$, consists of w subsets, in such a way that $\mathcal{S}^z = \mathcal{S}^z_1 \cup \mathcal{S}^z_2 \cup ... \cup \mathcal{S}^z_w$ and that for every i, j and z where $i, j \in \mathcal{W}$, $W = \{1, 2, ..., w\}$ and $z = \{0, 1\}$, it holds that $\mathcal{S}^z_i \cap \mathcal{S}^z_j = \emptyset$. $\mathcal{S}$ includes all shifts, independent of a shift's length, where $\mathcal{S}^z_1$, $\mathcal{S}^z_2$, ..., $\mathcal{S}^z_w$ represent the subsets of shifts where every subset has a distinct shift length and an information point associated with the subset. Let $\mathcal{A} = \{1, 2, ..., j\}$ be such that there exist j different activities. An activity is working at a filter. Off-duty time o, break time b, the briefing $briefing$ and walking time q are excluded in the set of activities. Off-duty time o is not part of the shift, while b, q and $briefing$ are part of the shift. Let $\mathcal{T}$ represent intervals during a day. As example, suppose that there are 48 intervals during a day (intervals with length half an hour). In this example, we have that $\mathcal{T} = \{1, 2, ..., 48\}$. The set $\mathcal{T}$ grows when the interval length gets smaller. Generally, let $\mathcal{T} = \{1, 2, ..., m\}$ be such that there are m intervals of equal length.

Let me define some other terminology used in this thesis. Walking time is specified as the time in minutes it takes to walk from a start point to the break room and from there to the destination. The amount of walking time depends on the start point and the destination. Walking time is not included in the break time and hence comes supplementary to the break time.

In this thesis, the following assumptions are made regarding our schedules.

- Every time a team starts at a filter, there is a minimum number of minutes that the team needs to stay there. This is referred to as minimum standing time T_{min}. This also implies that this minimum standing time needs to be met at the end of the shift. Hence, if there is a minimum standing time of thirty minutes, the last break should end at least thirty minutes before the end of the shift. Walking time is excluded in the minimum standing time and hence there should be sufficient extra space for walking time.
- In addition to a minimum standing time, there is also a maximum standing time T_{max} for which the same conditions hold.

- For some filters, demand for security teams at a specific time can be zero. During that time, no teams have to be present at that filter. Moreover, more teams than the required number of teams can be scheduled. This results in the residual teams being idle. The problem of idle teams is handled during a further planning stage that is not discussed in this thesis.
- There is a briefing at the beginning of every shift. This briefing takes about 15 minutes (including walking time to the filter). This briefing time is part of both the standing time and the shift time.
- There is a subset $\mathcal{T}_I^w$ of intervals $\mathcal{T}$ in which a shift of type $w \in \mathcal{W}$ can start.
- Teams can be repositioned to a different filter. A team can only be repositioned after the team had a break.
- Let $w_1 \in \mathcal{W}$ be a shift length with type w_1. Shift length w_1 is subdivided in two categories, such that $w_1 = w_{1a} \cup w_{1b}$. A shift of type w_1 has fixed starting times. Shift type w_{1a} has start time s_{1a}. Similarly, shift of type w_{1b} has start time s_{1b}.
- Let $w_2 \in \mathcal{W}$ be shifts with type w_2. Shifts of type w_{2a} can be subdivided in two categories, such that $w_2 = w_{2a} \cup w_{2b}$. Shifts of type w_2 have an interval of possible starting times. For types w_{2a} and w_{2b} these intervals are respectively I_{2a} and I_{2b}.
- Other shift lengths $w \in \mathcal{W} \backslash \{w_1, w_2\}$ have a fixed shift length but do not have requirements regarding start and end times.
- Shifts with length type $w \in \mathcal{W}$ have a minimum and maximum requirement of the number of teams that can work that type of shift at time $t \in \mathcal{T}$.
- Demand is known upfront. It differs for every day as it depends on the number of flights.
- Depending on the type of shift length $w \in \mathcal{W}$, there are requirements for a long break. If shifts of type w have shift length below threshold β_1, there is no requirement for a long break. If shift length is between β_1 and β_2, there is a long break requirement of b_1 minutes without interruption. If the shift length is above β_2, the requirement for a long break is b_2 minutes. Out of these b_2 minutes, b_1 minutes should be without interruption. In case a long break is included in a shift, b_1 minutes of the break are unpaid.
- The long break should be sufficiently far from the start and end time of the shift. What is deemed sufficient depends on the length of the shift. If shift length of shift type w is smaller than α minutes, the break needs to be at least ξ_1 minutes away from start and end time. If shift length of shift type w is larger than α minutes, the break needs to be at least ξ_2 minutes away from the start and end time of the shift.
- There is a maximum number of short breaks in a shift, which depends on the shift length $w \in \mathcal{W}$. Let SB^w be the parameter that represents the maximum number of short breaks for shift length type $w \in \mathcal{W}$.

- Between the two information points, a minimum amount of $\gamma\%$ of similar shifts from the solution of information point $z = 0$ needs to return for information point $z = 1$. Shifts are similar if the start interval of the shift from $z = 1$ is at most a δ-interval difference away from the shift from information moment where $z = 0$ and the shifts have equal length.

At this point, both the definition and mathematical notation of the shift have been introduced, as well as the requirements for a shift. As an illustration, I will provide a short example of how a shift might look. Let $i \in \mathcal{S}_w^z$ be a shift. Let $a_j \in \mathcal{A}$ be an operation. The length of a shift is dependent on the shift type. For the reason that we are considering an example, let shift i have length 22. Hence, a shift can be represented by a sequence of a_j, $j \in \mathcal{A}$, b, q and $briefing$. Off-duty time o is not included in the shift. Furthermore, let $T_{min} = 3$ and $T_{max} = 5$. In the example below the shift is printed using bold letters.

$$[o, o, o, \mathbf{briefing}, \mathbf{a_1}, \mathbf{a_1}, \mathbf{a_1}, \mathbf{q}, \mathbf{b}, \mathbf{q}, \mathbf{a_2}, \mathbf{a_2}, \mathbf{a_2}, \mathbf{a_2}, \mathbf{q}, \mathbf{b}, \mathbf{q}, \mathbf{a_1}, \mathbf{a_1}, \mathbf{a_1}, o, o] \tag{1}$$

2.1 Binary programming formulation

In this section, the binary programming (BP) formulation of the problem is established. The section is structured as follows. First of all, the variables of the model are defined. Subsequently, the BP formulation is discussed.

Let $x_i^{w,z}$ be our decision variable, in such a way that $x_i^{w,z} = 1$ if shift i of length type w is chosen during information moment (period) z and 0 otherwise. Period 0 is concerned with the demand from four weeks before, while period 1 belongs to last weeks demand. Let c_w be the cost (number of hours worked) for shift length $w \in \mathcal{W}$. Let entry (i, j, t) of the three-dimensional matrix A represent whether shift i covers activity j during time interval t. Moreover, let entry (i, j, t, w) of the four-dimensional matrix B represent whether shift i covers activity j during interval t for shift type w. Moreover, let d_j^t be the demand for activity t during time interval t. By the same token, let min_w and max_w represent the minimum and maximum amount of shifts of a certain length type w that can be chosen, in such a way that $max_w \geq min_w$ (if $min_w = max_w$, an exact number of shifts of length w has to be chosen). For period 1, additional constraints and parameters exist. Let P_{ip}^w represent whether shift i of set S_w^1 can be coupled to shift p of set S_w^0, where both shifts have shift type w. Furthermore, let parameter γ be the minimum percentage of similar shifts for $z = 1$ that have to return from $z = 0$.

The Binary programming formulation is then as follows.

$$\min \sum_{w \in \mathcal{W}} \sum_{i \in \mathcal{S}_w^z} C_w x_i^{w,z} + C_\nu \nu \tag{2}$$

$$\text{S.t.} \sum_{w \in \mathcal{W}} \sum_{i \in \mathcal{S}_w^z} A_{ij}^t x_i^{w,z} \geq (1-\nu) d_j^t, \qquad \forall j \in \mathcal{A}, \quad \forall t \in \mathcal{T} \tag{3}$$

$$\sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{A}} \sum_{i \in \mathcal{S}_w^z} B_{ij}^{t,w} x_i^{w,z} \geq min_w, \qquad \forall w \in \mathcal{W} \tag{4}$$

$$\sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{A}} \sum_{i \in \mathcal{S}_w^z} B_{ij}^{t,w} x_i^{w,z} \leq max_w, \qquad \forall w \in \mathcal{W} \tag{5}$$

$$x_i^{w,z} \in \{0,1\}, \qquad \forall i \in \mathcal{S}_w^z, \quad \forall w \in \mathcal{W} \tag{6}$$

For $z = 1$, additional constraints prevail:

$$\sum_{i \in \mathcal{S}_w^1} P_{ip}^w x_i^{w,z} \leq 1, \qquad \forall p \in \mathcal{S}_w^0 \tag{7}$$

$$\sum_{w \in \mathcal{W}} \sum_{p \in \mathcal{S}_w^0} \sum_{i \in \mathcal{S}_w^1} P_{ip}^w x_i^{w,z} \geq \gamma |P|, \tag{8}$$

The problem is either solved for period 0 ($z = 0$) or period 1 ($z = 1$). When $z = 0$, the BP problem consists of objective (2) and constraints (3)-(6), while for $z = 1$ the problem consists of objective (2) and constraints (3)-(8). Objective (2) minimises the number of hours necessary to fulfil the demand. In addition to the cost component C_w, a dummy component $C_\nu \nu$ is included such that a feasible solution is always found. cost C_ν is an upper bound on the costs such that it is costly to use dummy variable ν. The upper bound equals 1.5 times the number of hours demanded. Constraint (3) enforces that demand at filter j during time interval t is met. Constraints (4) and (5) ensure that a minimum and a maximum number of shifts of a specific length type w are chosen. For $z = 1$, additional constrains prevail. Constraint (7) couples shifts for $z = 1$ to shifts from the optimal solution for this instance for $z = 0$. Parameter P_{ip}^w has value 1 if the start interval of shift i of S_w^1 is within δ intervals of the start interval of shift p for $p \in \mathcal{S}_w^0$ and the parameter has value 0 otherwise. For every column of P_{ip}^w, only one entry in the column can be unequal to zero. Hence, a shift in period 1 can only be coupled to one shift of period 0. In case the column generation model wants to include the shift another time for a different coupling point, the shift has to be added again to the the problem with this different coupling point. Constraint (7) forces that for every shift p of the optimal solution from set $\mathcal{S}_w^0$, at most one shift from $i \in \mathcal{S}_w^1$ can be conjugated. Lastly, constraint 8 enforces that γ per cent of similar shifts come back in future the solution for $z = 1$, where $|P|$ is the number of shifts in the optimal solution from $\mathcal{S}^1$.

3 Literature

The literature review is outlined as follows. First, the single-activity shift scheduling problem (SASSP) and crew scheduling problem (CSP) are discussed. The multi-activity shift scheduling problem (MASSP) considered in this thesis is an extension of the SASSP. Therefore, the foundation of both problems is very similar and hence the SASSP is a good starting point for our literature review. Subsequently, literature regarding the MASSP is investigated. Lastly, literature regarding the multi-activity tour scheduling problem, an extension of the multi-activity shift scheduling problem, is reviewed.

Shift scheduling problems have been around since the work of Edie (1954) and the set covering solution approach to his work by Dantzig (1954). As shift/crew scheduling problems are NP-hard, researchers need to resort to other methods to solve large real-life instances such as the problem considered in this thesis. For an overview of both the SASSP and the CSP, the author would refer the reader to Ernst et al. (2004) (general overview), Heil et al. (2020) (railway crew scheduling) and Kasirzadeh et al. (2017) (airline crew scheduling) who provide a complete overview of the existing literature. Methods for dealing with the SASSP include for instance column generation (Brunner and Edenharter (2011)) and metaheuristics (Maenhout and Vanhoucke (2007); Aickelin and Dowsland (2004)), but other methods exist. Approaches for the CSP include column generation (Desrochers and Soumis (1989); Borndörfer et al. (2006); Abbink et al. (2011)) and metaheuristics (Lourenço et al. (2001); Hanafi and Kozan (2014)). Research regarding security scheduling at airports is limited. Examples include Soukour et al. (2012) and Soukour et al. (2013). The authors apply a construction/destruction and memetic algorithm, respectively, to a staff assignment problem. This is different from this research because the authors researched a different step in the scheduling process. Their focus is on the assignment of shifts, while this research concerns the creation of shifts. Moreover, Soukour et al. (2012) and Soukour et al. (2013) do not consider crossovers between different activities.

The more complicated multi-activity shift scheduling problem (MASSP) is an extension of the SASSP. The SASSP is an NP-hard problem, and consequently, so is the MASSP. In the MASSP, one is determining where and at which time a set of employees should carry out a job. Research regarding the MASSP is relatively scarce. Existing research focuses mainly on three approaches: regular and context-free languages, column generation and heuristics.

Regular languages are more common in literature discussing the multi-activity shift scheduling problem. Salvagnin and Walsh (2012) use regular languages for a hybrid MIP/CP approach. The regular languages are thereby used to partially describe the set of feasible shifts. The authors show that their method is comparable to current state-of-the-art methods for a small number of activities. The method becomes less competitive as the number of activities increases. Moreover, the authors state that the method significantly outperforms explicit MIP-based formulations.

Quimper and Rousseau (2010) use regular and context-free languages to model the scheduling rules.

This results in specialised graph structures. These structures can be efficiently searched using large neighbourhood search. This method often results in near-optimal solutions for the SASSP and also tends to perform well for the MASSP, solving instances with up to ten activities. Côté et al. (2013) use a branch-and-price algorithm to solve the personalised version of the MASSP, in which the subproblems are, similar to the method in Quimper and Rousseau (2010), formulated using context-free grammars. The authors mention that a grammar-based column generation approach is generic and therefore can handle a variety of shift problems. They show that their approach for personalised problem instances is comparable with a specialised heuristic method and even superior to this heuristic in some cases. Côté et al. (2013) also remark that a large set of rules can be modelled by using context-free grammars. However, they also observe that context-free grammars are hard to use when for instance the start and end times of shifts are not fixed upfront. Therefore, as our starting and end times are not fixed, the use of context-free grammars might not be optimal for the research considered in this thesis.

Dahmen et al. (2018) propose an implicit formulation using forward and backward constraints to solve the MASSP. They conclude that their approach was able to solve large instances within a reasonable time. This is because the number of constraints does not exponentially grow with the number of activities in their model. Their approach to handling their pre-break and post-break profiles results in an exhaustive enumeration of these profiles. On the one hand, this is a limitation, but on the other hand, it results in freedom regarding the modelling of complex rules.

Restrepo et al. (2012) use a column generation-based approach, with as an auxiliary problem a shortest path problem with resource constraints (SPPRC). They test their approach on data from a parking lot operator in Bogotá, Colombia. They were able to efficiently achieve a near-optimal solution to problems with up to sixteen work activities over a one-week planning horizon. They show a considerable reduction in costs and required man-hours if the operator allows staff to transfer between different lots within shifts. This is of interest to our problem as well, as this implies that we could reduce our costs by allowing people to transfer from one filter to another. The method presented in Restrepo et al. (2012) has advantages, as specialised algorithms exist to solve the SPPRC. An example of such a specialised algorithm is proposed in Lozano and Medaglia (2013). Relatively large problems can hereby be solved exactly within a reasonable time. In Lozano and Medaglia (2013), a column generation method embedded with integer programming is compared to a column generation approach with a pulse algorithm, and this shows a substantial decrease in solution time. The research considered in this thesis is different from the research by Restrepo et al. (2012) as there are different scheduling rules. Moreover, there are different shift lengths possible in our research compared to the standard 8 hours in their research. Moreover, overtime is allowed in Restrepo et al. (2012).

Related research is performed by Al-Yakoob and Sherali (2008). They consider an employee scheduling problem faced by the Kuwait National Petroleum Company in managing their gas stations. To solve this problem, they use a column generation heuristic (CGH) that was able to solve the problem

up to 90 gas stations and 336 employees. As a heuristic, they implemented a sequential variable-fixing heuristic. The authors show that, compared to the two-stage procedure (TSA) presented in Al-Yakoob and Sherali (2007), they were able to solve considerably bigger problems (the procedure in Al-Yakoob and Sherali (2007) solves problems up to 29 stations and 84 employees). Moreover, the CGH showed between 6 and 33 per cent better solutions than the TSA. The research in Al-Yakoob and Sherali (2008) is different compared to Restrepo et al. (2012) as there are only three scheduling options in the former, while in the latter any time could be a starting time. This is also a difference compared to this research.

Pan et al. (2016) apply an ILP formulation to find a solution to a problem where only workload constraints are considered and then extend the solution using a Tabu search approach by also including constraints regarding the maximum and minimum duration of activities. Other legal regulations are not taken into account. They compare their approach to a binary linear programming formulation. They find feasible solutions within a reasonable time with their approach. To the same problem, CPLEX does not always find a solution, especially when the size of the problem increases. The authors remark that there is a big impact of applying intensification and diversification in their algorithm (increasing the percentage of solved instances from 50% to respectively 67 % and 83%).

The existing literature shows us that, depending on the method, large instances of the multi-activity shift scheduling problem can be dealt with. Two methods are more frequently used. The first one is an approach using context-free grammars, while the other one is a column generation approach. Both approaches have their advantages and disadvantages. The context-free grammars provide a lot of flexibility regarding the incorporation of decision rules, but some rules are tough to implement by using context-free grammars. An example is scheduling over a longer period when starting and ending times of shifts are not prearranged (Côté et al. 2013). For the column generation approach, the network structure of the pricing problem can facilitate a smooth implementation of more difficult constraints. However, one of the disadvantages of using a network structure is that the size of the problem grows exponentially when smaller time intervals are utilised. To illustrate this, consider a five-minute interval compared to a thirty-minute interval. To model a full day of work, a total of 288 nodes needs to be combined in case of the five-minute interval while for the thirty-minute interval only 48 nodes need to be combined. This has a considerable effect on computation time. In addition to exact approaches, heuristics are also used in the pricing problem. The column generation approach seems, according to the aforementioned literature, to be able to handle more activities than the context-free grammar approach, but one should make the remark that not all mentioned research considers the same time limit and computational power for their research.

An extension of the MASSP is the multi-activity tour scheduling problem (MATSP). The MATSP is the MASSP combined with days-off scheduling. As shift scheduling is part of the MATSP, similar computational complexities arise as in the MASSP. Moreover, some research also partially determine the computational efficiency of their method by using problem instances of the MASSP.

An example of such a study is Gérard et al. (2016). Their research is also interesting as they have time-varying demand - similar to this research. Compared to this research there are two main differences. First of all, their research also incorporates days-off scheduling. Secondly, their objective is to minimise the amount of over and under coverage, while in this research under coverage is not allowed. They show that methods based on Dantzig-Wolfe decomposition provide good results.

Restrepo et al. (2018) propose a Benders decomposition method, in which the variables corresponding to tour patterns are used in the master problem and are linked with variables related to daily shifts. They solve the Benders master problem using column generation. The subproblems are solved using context-free grammars. The authors test their approach on real-world instances for both the MATSP and the MASSP. For the MATSP high-quality solutions are found up to ten instances, while for the MASSP, the Benders decomposition approach solves instances up to 30 work activities and 137 employees within 1 % optimality.

Qu and Curtois (2020) provide a set of benchmark problems for the MATSP, with varying planning lengths, as well as a varying number of activities and staff available. A variable neighbourhood search approach is implemented to solve the problems. Small instances with up to seven days and a few activities and staff are solved within five minutes, while for some larger problems even a feasible solution could not be found in ten minutes.

The research in this thesis is distinctive from the existing literature for the following reasons. In this research, there are no strict starting times. Therefore, the pattern of shifts is irregular. Moreover, there are several different shift lengths, with each their own requirements. Existing literature often considers regular shifts patterns (e.g. a day, evening and night shift) and no diversion in shift lengths. In this thesis, both are taken into account. Hence, to the best of the author's knowledge, this research is original and relevant as addition to the existing literature.

4 Methodology

In this section, a further elaboration of the methods applied in this thesis is given. The binary programming formulation is already provided in Section 2. This section describes the column generation approach and further methods used in this thesis. The section is structured as follows. First, the column generation algorithm is introduced in Section 4.1. Secondly, the initialisation algorithm is discussed in Section 4.2. Thirdly, the reduced costs are discussed in Section 4.3. Fourthly, the exact (Section 4.4) and heuristic (Section 4.5) algorithms are discussed. Lastly, the method to translate the column generation solution to an integer solution is discussed in Section 4.6.

4.1 Column generation

The BP formulation (2)-(8) is solved using a column generation approach. In this section, a generic overview of the column generation method is provided. The method needs the following input: the demand d_j^t, $\mathcal{W}$, min_w and max_w, $initialShifts$, $MaxTimeLimit$ and $maxWithoutImprovement$. These inputs represent the demand, the set of shift types, the minimum and maximum amount of shifts of each type, the set of initial shifts, the maximum time limit and the maximum iterations without improvement. The algorithm can be found in Algorithm 1.

Algorithm 1 Column generation

procedure COLUMN GENERATION(d_j^t ($\forall j \in \mathcal{A}, \forall t \in \mathcal{T}$), $\mathcal{W}$, min_w ($\forall w \in \mathcal{W}$), max_w ($\forall w \in \mathcal{W}$), initialShifts, maxTimeLimit, maxWithoutImprovement)

Step 1: Initialize Restricted Master Problem (RMP) with a small set of feasible shifts (see Section 4.2) such that the RMP provides a feasible solution.

while $time <$ maxTimeLimit **and** $noImprovement <$ maxWithoutImprovement **do**

for $w \in \mathcal{W}$ **do**

Step 2: Solve the RMP(d_j^t, min_w, max_w). ▷ $\forall j \in \mathcal{A}$, $\forall t \in \mathcal{T}$, $\forall w \in \mathcal{W}$

Step 3: Solve the pricing problem exactly (see Section 4.4) or heuristically (see Section 4.5) for shift length w. If any shifts with negative reduced costs are found, proceed to step 4. Else go to step 5.

Step 4: Add the shifts that follow from step 3 to the RMP. Set $noImprovement$ to 0. Return to step 2 for next $w \in \mathcal{W}$.

Step 5: If stopping criteria reached: break out of for-loop. Else: proceed to step 6.

Step 6: $noImprovement = noImprovement + 1$. Continue to next $w \in \mathcal{W}$.

Step 7: The solution to the LP relaxation is found.

Return: solution to the LP-relaxation $x^\star$

The column generation algorithm (CG) initialises with a small, feasible set of shifts. In the Master Problem (MP), these variables are binary. At the same time, these variables are relaxed in the Restricted Master Problem (RMP). Hence, the RMP is the relaxation of the BP formulation as described in Section 2.1 and is solved in step 2. By duality theory, the RMP provides us with the dual values of the constraints. The dual values are then utilised to construct the objective in the pricing problem (see Section 4.3). The pricing problem is solved using this objective with either an exact or heuristic pricing problem. If any shifts with negative reduced costs are found, these are added to the RMP. Subsequently, the RMP is solved. If no shifts are found, the $noImprovement$ parameter is increased by one. This process is repeated until either the time limit is reached or no shifts with negative reduced costs are found for at least $maxWithoutImprovement$ iterations. At that point, the solution to the LP relaxation, $x^\star$, is found. To get to a final result for the problem at hand, the solution of the column generation algorithm has to be made integer. The process for

this can be found in Section 4.6.

4.2 Initialisation

For the initialisation of the column generation algorithm, a feasible set of shifts has to be created. The procedure described in this section consists of three components. In the first component, a small set of starting shifts is created. In the second component, this set of starting shifts is used in combination with data for a week to create a larger set of shifts. In this way, the set of shifts includes day-specific patterns. In the last component, during the start of the column generation procedure, the restricted master problem is solved and the shifts that are chosen in the optimal solution to the RMP are taken from the large set of shifts. Moreover, some of the shifts that are not included in the optimal solution are included in the RMP, to increase the number of possibilities for the column generation.

Let us consider the components in their natural order of application. For the first component, let us start by defining sets and parameters. Define the set $\mathcal{T}_I^w$ as the set of starting intervals in the initialisation procedure. This set is a subset of all intervals $\mathcal{T}$ and is dependent on the shift length w, as the possible starting intervals depend on the shift length type w. Moreover, let M_{SD}^w be the maximum number of shift divisions created for a certain shift type $w \in \mathcal{W}$. Let SD^w be the set of shift divisions of shift type $w \in \mathcal{W}$. A shift division is a set of activity lengths such that the sum of the activity lengths plus the required breaks equals the required shift length. Moreover, let *lengthsActivities* represent the time of an activity, such that the length is between the minimum and maximum standing time. In case the problem is solved for period 1, the initial set of shift consists of the outcomes for the same instances for $z = 0$.

Algorithm 2 Initialisation Algorithm: component 1

1: **procedure** INITIALISATION ALGORITHM: COMPONENT 1($\mathcal{T}_I^w$, $\mathcal{W}$, M_{SD}, *lengthsActivities*)
2: Initialisation: *startingShifts*[], SD^w[] ($\forall w \in \mathcal{W}$)
3: **for** $w \in \mathcal{W}$ **do**
4: **for** i in M_{SD} **do**
5: **if** w needs break **then**
6: StartTime$_b$ = $t_{i,b}$ ▷ Determine start time break
7: EndTime$_b$ = $t_{i,b}$ + *lengthLongBreak* ▷ Determine end time break
8: LABB = random(lengthsActivities) ▷ LABB = lengths activities before break
9: LAAB = random(lengthsActivities) ▷ LAAB = lengths activities after break
10: $SD^w[i]$ = createShiftDivision(LABB, LAAB, $startTime_b$, $endTime_b$)

11: **for** every $w \in \mathcal{W}$ **do**
12: **for** every t_i in $\mathcal{T}_I^w$ **do**
13: shift = []
14: StartTime = t_i
15: EndTime = t_i + length(w)
16: **for** i in NOSPST **do** ▷ NOSPST = number of shifts per start time
17: shiftDivision = random(SD^w)
18: **for** activityLength in shiftDivision **do**
19: activityChosen = random($\mathcal{A}$)
20: **for** activity in activityLength **do**
21: shift.append(activityChosen)
22: **if** breakNeeded **then**
23: **for** break in breakLength **do**
24: shift.append(break)
25: shift, feasible = adjustWalkingTime(shift)
26: **if** feasible **then**
27: startingShifts.append(shift)
28: **Return**: startingShifts

At this point, let us discuss the algorithm described in Algorithm 2. First of all, in lines 3-10, a set of shift divisions is created. A shift division includes the activities, short and long breaks and the walking time. The short breaks can be determined from the LABB en LAAB, as between the activities a break needs to be held. The long breaks are situated between the LABB and LAAB. This part of the process starts by determining during which interval the long break starts. From there, the length of the activities before and after the break are determined. At this point, no information is known about the activities within the shifts. Moreover, the maximum walking time is included

for both the short and long breaks to make sure any combination of activities is feasible. In the second part of the the algorithm, lines 11-27, the created shift division are paired to a starting and ending time. Moreover, the activities within the shift are decided upon. The walking time is adjusted for the activities while preventing any violations of the rules and regulations such as infeasible walking time. Making sure that a feasible starting solution is created when shift lengths are not infinite, is already a $\mathcal{NP}$-hard problem. Therefore, a dummy variable ν is included in constraint (3) of the BP formulation. This variable makes sure that a feasible solution always exists for the second component (although it is a relative expensive variable to use during the second component).

At this point, the first component is executed and the second component can be initialised using the results of the first component. In the second component, a large set of feasible shifts is created by using the column generation algorithm and data for a week. This component uses the column generation with an exact pricing problem. By applying this method, one can account for day-specific patterns within the data. The pseudo-code for the second component of the initialisation algorithm can be found in Algorithm 3.

Algorithm 3 Initialisation Algorithm: Component 2

1: **procedure** INITIALISATION ALGORITHM: COMPONENT 2(startingShifts, trainingWeek, maxTimeLimit, maxWithoutImprovement)
2: **for** trainingDay in trainingWeek **do**
3: totalShifts = startingShifts
4: shifts = ColumnGeneration(trainingDay, $\mathcal{W}$, min_w, max_w, startingShifts, maxTimeLimit, maxWithoutImprovement)
5: totalShifts.append(shifts)
6: **Return:** totalShifts

In Algorithm 3 a large set of initial shifts is created. This initialisation set can now be used for the daily column generations. The process for this is described as follows.

Algorithm 4 Initialisation Algorithm: component 3

```
1: procedure Initialisation Algorithm: Component 3(totalShifts)
2:     Initialisation: shiftsNew[]
3:     dualValues, solutionValues = RMP(totalShifts)
4:     for shift in totalShifts do
5:         if solutionValues(shift) > 0 then
6:             shiftsNew.append(shift)
7:         else:
8:             randomChoice = random(0, 1, φ)    ▷ 0 with probability 1 − φ, 1 with probability φ
9:             if randomChoice = 1 then
10:                shiftsNew.append(shift)
11:    Return: shiftsNew
```

In Algorithm 4, step 1 of Algorithm 1 is described. In line 3 of the algorithm the RMP is solved. Subsequently, in the lines (4)-(10) the starting set of shifts is determined. There are two parts of this. First of all, part of the shifts are part of the optimal solution of the RMP (these shifts have solution values in the range $(0, 1]$). Additionally, the set of shifts is expanded by randomly including additional shifts. A shift is randomly included with probability ϕ and not included with probability $1 - \phi$.

The first and second components of the initialisation algorithm only have to be created once for a certain number of filters. The third component, as described in Algorithm 4, is employed every time the column generation method from this thesis is used (step 1 of Algorithm 1).

4.3 Reduced costs

In this section, the reduced costs of the pricing problem are constructed. Moreover, the cost of visiting a node is clarified. First of all, every time the solution to the RMP has been found, a new column to be included in the problem formulation has to be created or we have to show that no columns with negative reduced costs exist. This new column represents a shift that copes with all the requirements to be a valid shift. This shift is created using an exact pricing algorithm (see Section 4.4) or a heuristic pricing algorithm (see Section 4.5). These algorithms are based on finding shifts with negative reduced costs. The reduced costs of shift follows from the constraints of the restricted master problem. More precisely, let λ_j^t, π_w, κ_w, μ_p, ψ be the negative dual variables that correspond with constraints (3), (4), (5), (7), (8), respectively. The reduced cost is dependent on z. As mentioned, for $z = 0$, constraints (7) and (8) are not included in the formulation. Therefore, the reduced costs for a shift when $z = 0$ equal

$$RC(x_i^{w,0}) = C_w + \sum_{j \in \mathcal{A}} \sum_{t \in \mathcal{T}} A_{ij}^t \lambda_j^t + \pi_w - \kappa_w \tag{9}$$

For $z = 1$, the reduced costs are given by

$$RC(x_i^{w,1}) = C_w + \sum_{j \in \mathcal{A}} \sum_{t \in \mathcal{T}} A_{ij}^t \lambda_j^t + \pi_w - \kappa_w - \sum_{p \in \mathcal{S}_w^0} \mu_{pw} P_{ip}^w + \psi \tag{10}$$

In the pricing problem, the reduced costs are minimised.

4.4 Exact pricing algorithm

The exact approach to solve the subproblems uses a shortest path problem with resource constraints (SPPRC). The SPPRC is an $\mathcal{NP}$-hard problem. Nonetheless, fast, specialised algorithms to solve the SPPRC exist. In this section, the methodology behind solving the SPPRC is reviewed. This section is structured as follows. First, the problem is translated into a network structure. Subsequently, the resources of the SPPRC are discussed. Finally, an extensive elaboration of the exact pulse algorithm is included in this section. The objective of the algorithm discussed in this section is finding the most valuable shifts out of the set of shifts $\mathcal{S}$. The value of a shift is based on the reduced costs that such a shift yields. A shift is most valuable if it has the most negative reduced costs. Every node on the optimal path $P^\star$, that together represents a shift, yields some cost. This cost follows from the dual variables of the master problem. The path $\mathcal{P}$ starts at node s and ends at node t and has a prespecified length L, which is the length of the shift. Before clarifying the cost structure more, let us first consider the network.

Network structure

The SPPRC is modelled using a network structure. Let us define this formally. Let $G = (\mathcal{V}, \mathcal{E})$ be a directed acyclic graph with $\mathcal{V}$ as the set of nodes and $\mathcal{E}$ the set of arcs. The graph G is directed since G is a graph in a time-space network. Hence, no cycles exist. The nodes represent working at an activity during a specific time interval. An arc from node i to node j represents a feasible opportunity to traverse from node i to node j. If no arc exists, no direct feasible path exists between node i and node j. To increase understanding, one can take a look at Figure 2. In the figure, the nodes represent working at filter j during time interval t (t can be found in the top bar). One might notice that before t equals MINST (minimum standing time) no arcs exist between activities. A worker is not allowed to switch activities if he/she has not been working that activity for at least the minimum standing time. Every time a break has been held, the time variable resets and a similar pattern occurs for $time < MINST$. Blue lines indicate break and walking time variables. As the walking time depends on the start and end point, the total amount of time for walking and having a break differs per combination of activities. To keep Figure 2 of reasonable size, only breaks ending at filter 1 are included. Moreover, only short breaks are included in the network formulation of Figure 2. Long breaks work similarly as short breaks but take longer. These breaks are not included in Figure 2. For all other filters, the same blue lines could be drawn which result in similar patterns for the short breaks. Moreover, in practice, there is a whole subset of starting

points T_I^w and all these start points have a similar network.

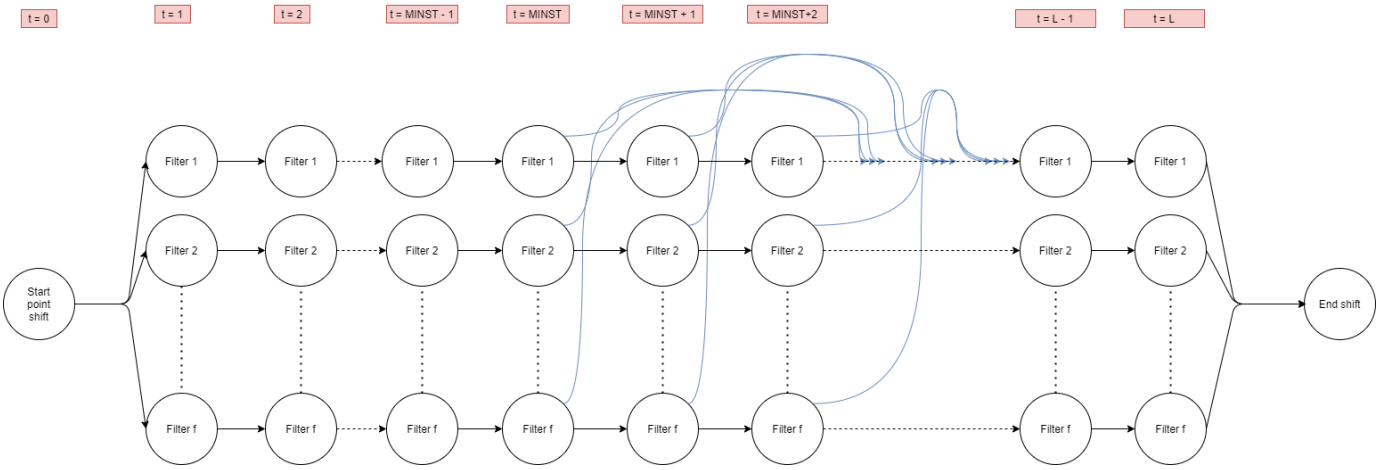

Figure 2: Example of a network

Every arc from node i to node j in the network has a cost c_{ij} associated with the arc. Recall that the set of nodes is given by $\mathcal{V}$. This cost c_{ij} follows from the reduced cost of the problem. Let node i be represented by $i = (f_i, t_i) \in \mathcal{V}$, where f_i is the filter of node i and t_i is the interval of node i. In addition, there are several extra nodes: start point s, $briefing$ and end point t. These have no specified interval in which they occur. The cost of traversing an arc from node i to node j for $z = 0$ is given in (11). For $z = 1$, the cost of traversing from node i to node j is given in (12).

$$C_{ij} = \begin{cases} 0 & \text{if } i = (s, -) \text{ and } j = (\text{ briefing}, -) \\ C_w + A_{ij}^t \lambda_j^t + \pi_w - \kappa_w & \text{if } i = (\text{ briefing}, -) \text{ and } j = (f_j, t_j), \text{ for } j \in \mathcal{V} \\ A_{ij}^t \lambda_j^t & \text{if } i = (f_i, t_i) \text{ and } j = (f_j, t_j), \text{ for } i, j \in \mathcal{V} \\ 0 & \text{if } j = (t, -) \end{cases} \tag{11}$$

$$C_{ij} = \begin{cases} 0 & \text{if } i = (s, -) \text{ and } j = (\text{ briefing}, -) \\ C_w + A_{ij}^t \lambda_j^t + \pi_w - \kappa_w - \sum_{p \in \mathcal{S}_w^0} \mu_{pw} P_{ip}^w + \psi & \text{if } i = (\text{ briefing}, -) \text{ and } j = (f_j, t_j), \text{ for } j \in \mathcal{V} \\ A_{ij}^t \lambda_j^t & \text{if } i = (f_i, t_i) \text{ and } j = (f_j, t_j), \text{ for } i, j \in \mathcal{V} \\ 0 & \text{if } j = (t, -) \end{cases} \tag{12}$$

Resources

The SPPRC has restrictions regarding resources. Mathematically, let $\mathcal{R} = (R_1, R_2, ..., R_r)$ be a set of r resources. The shortest path in a network is feasible if and only if all resources are satisfied. In the model discussed in this thesis, several resources exist. Let R_1 be the resource that measures the standing time. There is both a minimum and a maximum standing time, respectively T_{min} and T_{max}. The amount of time a team works at an activity before switching to another activity should be between T_{min} and T_{max}. Secondly, resource R_2 measures whether the requirement for a long break that is sufficiently long is satisfied. In case the break time is split up into two parts, the component with the longest length, b_1, is taken as the rest time. Resource R_3 measures the time since the last unpaid moment; this could be either the start of the shift or the long break. Resource R_4 counts the short breaks.

Algorithm

To solve the SPPRC, Lozano and Medaglia (2013) propose a fast algorithm based on sending pulses through a network. The algorithm provided in their research is also implemented in this thesis. The pseudocode of the algorithm can be found in Algorithm 5. Although the algorithm is straightforward, it is further clarified after the introduction of the algorithm. First, let us define some parameters. Let L be the required shift length and let $\mathcal{P}$ be the path. Define $\mathcal{P}^\star$ as the shortest, resource-feasible path through the network. Moreover, let C be the current cost of the path and let $\mathcal{R}$ be the current vector of resources. Finally, let v_k be the current node that is added to the pulse. The algorithm is initialised by starting the algorithm at start point s.

Algorithm 5 Pulse Algorithm

1: **procedure** PULSE ALGORITHM(L, $numberActivities$)
2: $\mathcal{P} = \{\}$
3: $C = 0$
4: $G_L = createGraph(L, numberActivities)$
5: $\mathcal{R} = (0, 0, 0, 0)$
6: $v_k = s$
7: pulseFunction(L, $\mathcal{P}$, C, $\mathcal{R}$, v_k)
8: **Return**: Optimal path $\mathcal{P}^\star$

The input of the algorithm is the shift length L for which the algorithm should produce a shift. Depending on the shift length, the graph related to this shift length is created in line 4. The graph depends on the shift length as the shift length has a large impact on the feasibility of some combinations as well as the requirements of a shift. Since computational efficiency is of the utmost interest, letting the graph depend on the shift length results in generating smaller graphs. Smaller graphs tend to take less time, as there the number of paths to go through is smaller. The algorithm

proceeds by calling the pulse function in line 7. The pulse function is described in Algorithm 6. The optimal path $\mathcal{P}^\star$ follows from the best bound of the pulse function.

At this point, let me shortly illustrate the reasoning behind the pulse algorithm. In the pulse algorithm, pulses are sent through a network G. The pulse function thereby sends pulses through the network. As one can imagine, even for small-sized networks, the number of possible paths is enormous. The pulse algorithm enumerates through all these options. However, not every path is resource-feasible. Therefore, in this thesis, three aggressive pruning strategies are applied. These pruning strategies help to efficiently prune large parts of the network at the point where it is already clear that a path is not feasible or better than the best path we have found until that point. The pruning strategies are based on feasibility, bounds and dominance. Let me clarify the pruning strategies here.

The first pruning strategy is based on feasibility. For the resource vector $\mathcal{R}$, if any of the resources of resource vector $\mathcal{R}$ violate a restriction or if a feasible solution is unattainable, the path can be pruned. As there are multiple resources in this research, feasibility pruning is very powerful. Secondly, pulses are pruned by bounds. This is most easily interpreted as follows. The pulse algorithm uses the deepest node first selection strategy. Once we have reached the final node t for the first time, this results in an upper bound. For any other path that passes through the algorithm at a later stage, we know that, if the path does not have the potential to reach a lower solution than the upper bound, the path can be pruned. The deepest node first selection strategy is particularly interesting for the pulse algorithm, as a quick upper bound on the objective can be found. This upper bound can be used to quickly prune paths from the network that result in worse solutions than the upper bound. To be precise, let C_{ub} be the current upper bound objective. If a solution $C(\mathcal{P})$, where $\mathcal{P}$ is a resource-feasible path (which might not be a full path from s to t) is found and there is no possibility to find a path with lower costs, this path can be pruned. At the occasion that a tighter upper bound is found, the upper bound can be updated. Since the upper bounds of paths can be updated, this directly shows the reasoning behind the deepest node first strategy. The last pruning strategy is based on the dominance of paths. Suppose we have path 1 with resource vector $\mathcal{R}_1$ with solution value c_1 and path 2 with resource vector $\mathcal{R}_2$ with solution value c_2, both being paths to node v. If $c_1 < c_2$ and $\mathcal{R}_1 = \mathcal{R}_2$. We then know that path 1 strictly dominates path 2 and that path 2 can be discarded.

Now, let us define the pulse function as mentioned in Algorithm 5. The pulse function is described in Algorithm 6.

Algorithm 6 Pulse function

```
1: procedure PULSE FUNCTION(L, 𝒫, c_𝒫, ℛ, v_k)
2:     output: void
3:     if feasible = True then
4:         if Bounds = True then
5:             if noDominance = True then
6:                 𝒫_p = 𝒫 ∪ v_k
7:                 for each outgoing arc v_i from node v_k do
8:                     c_p = c_𝒫 + c_ki
9:                     ℛ_p = ℛ + ℛ_{v_ki}
10:                    pulseFunction(L, 𝒫_p, c_p, ℛ_p, v_i)
```

At this point, let me shortly illustrate the pulse function. The pulse function is initialised by feeding the length of the shift L, the current path $\mathcal{P}$, the costs c_p and the set of restrictions $\mathcal{R}$ and the current node. First, in lines 3-5 it is determined whether adding this node to the path results in a resource-feasible path, whether this path can result in a path with a lower reduced cost and whether the path created is not dominated by any other path. If the checks all return true, the node can be added to the path. Next, in line 7 all subsequent nodes that are connected by an arc to this node (and hence might be feasible) are considered. For lines 8 and 9 the cost and resource vector are updated. Lastly, in line 10 the pulse function is called again and the procedure starts over. The procedure ends at the point where there are no subsequent nodes for any node. At this point, the best solution is determined and the bounds are updated. The function repeats itself until all feasible paths are examined.

To summarize, the pulse algorithm is always able to find the optimal path in this SPPRC if there exists one since the algorithm enumerates through all options. Options that are not feasible are pruned and therefore any further additions to these paths can be ignored. Consequently, large parts of the networks are cut off which improves the speed of the algorithm. For large networks, the algorithm still has to enumerate through numerous options. Therefore, no guarantee regarding the speed of the algorithm can be found and the algorithm is very dependent on the quality of the pruning strategies.

4.5 Heuristic pricing algorithm

In addition to the exact pulse algorithm, a heuristic algorithm is implemented to solve the problem. Heuristic algorithms tend to provide results more quickly at the cost of optimality. Hence, for the paths found with the heuristic algorithm, no guarantee exists regarding the quality of the paths. Moreover, the heuristic in this section is not dependent on other pruning strategies than feasibility. The heuristic algorithm in this section uses the initial solution from component 3 of Section 4.2 as the basis. For the algorithm described here, let the set of initialShifts be called $\mathcal{I}$. A shift i in

$\mathcal{I}$ consists of a starting time, end time, the activity times within a shift and at which activity the employee should be at what time. Moreover, the shift length L is known. Shift i can be sliced into small shift sections that represent working at an activity and a short or long break afterwards, except for the first and last slices of the shift. The first and last slices are different since the first slice has a briefing at the beginning and the last has no break at the end. For the heuristic method, a similar cost function can be utilised as in the exact method, but there are two differences. First, in the exact method, the costs are counted while the shift is created, while in the heuristic method, this only happens after the shift is created. Second, the shift does not have a specific start time upfront and therefore the costs are determined afterwards. Let us first consider the algorithm, after which we can determine the costs.

The algorithm can be found in Algorithm 7.

Algorithm 7 Heuristic algorithm

```
1: procedure HEURISTIC ALGORITHM(I, L, T_I^w, maxNumberIterations, maxTime, maxShifts)
2:     initialisation: bestObjective = ∞, components, numIterations = 0, optimalShifts[], profiles

3:     while len(newShifts) < maxShifts and time < maxTime do
4:         profile = selectProfile(profiles)
5:         startComponent = components(profile)
6:         shift = startComponent
7:         lengthRemaining = L - length(startComponent)
8:         for action in profile[1:] do
9:             couplingPossibilities = selectPossibleComponents(action, lengthRemaining)
10:            if len(couplingPossibilities > 0 then
11:                component = random(couplingPossibilities)
12:                lengthRemaining = lengthRemaining - length(component)
13:                shift = shift + component
14:                shift, feasible = adjustWalkingTime(shift)
15:                if feasible is false then
16:                    infeasible = 1
17:                    break
18:            else
19:                infeasible = 1
20:                break
21:        if infeasible = 1 then
22:            Continue
23:        for starting time s in T_s do
24:            reducedCost = cost(newShift)
25:            if reducedCost < 0 then
26:                newShifts.append(newShift, startTime)
27:    return: newShifts with negative reduced costs of shift length L
```

Algorithm 7 is quite straightforward. The process is initialised by determining the components from the shifts. A component is defined by the type of the component, the working time and the total length of the component. The type of component is determined by whether a briefing is included, whether a break is included and which kind of break. In total, five types of components exist. Every possible length of shift L has certain requirements regarding characteristics. For example, there is a maximum number of short breaks. The profile of the created shift is determined up front, in line (4). The profile informs about the types of components present in the shift. The first component of the shift is determined. From there, the following procedure is followed. First of all, from the set of components the subset of all feasible components is determined based on the remaining length and

the positioning of the action in the profile. From this subset, a component is randomly selected. The activities in a component are then appended to the existing shift and, if necessary, the walking time is adjusted to have the right length. In case the newly created shift is not possible - for example, if both components already work as much as the maximum standing time and there is a surplus of walking time, which can therefore not be included in either one of the components, the shift is infeasible. If this is the case, the process is restarted. Once a feasible shift is found, the cost of starting this shift at a specific time is determined. At this point the costs can be determined, as the starting point, the activities and their start and end times are known. The nodes through which the path goes, have a certain cost and these costs can be determined. The cost for a single node is as follows.

$$C_{ij} = \begin{cases} 0 & \text{if } i = (s, -) \text{ and } j = (\text{ briefing}, -) \\ C_w + A^t_{ij}\lambda^t_j + \pi_w - \kappa_w & \text{if } i = (\text{ briefing}, -) \text{ and } j = (f_j, t_j), \text{ for } j \in \mathcal{V} \\ A^t_{ij}\lambda^t_j & \text{if } i = (f_i, t_i) \text{ and } j = (f_j, t_j), \text{ for } i, j \in \mathcal{V} \\ 0 & \text{if } j = (t, -) \end{cases} \tag{13}$$

$$C_{ij} = \begin{cases} 0 & \text{if } i = (s, -) \text{ and } j = (\text{ briefing}, -) \\ C_w + A^t_{ij}\lambda^t_j + \pi_w - \kappa_w - \sum_{p \in \mathcal{S}^0_w} \mu_{pw} P^w_{ip} + \psi & \text{if } i = (\text{ briefing}, -) \text{ and } j = (f_j, t_j), \text{ for } j \in \mathcal{V} \\ A^t_{ij}\lambda^t_j & \text{if } i = (f_i, t_i) \text{ and } j = (f_j, t_j), \text{ for } i, j \in \mathcal{V} \\ 0 & \text{if } j = (t, -) \end{cases} \tag{14}$$

In case the total costs of a shift are negative, the shift is included in the RMP. The heuristic provides a set of shifts that can are added simultaneously.

4.6 Integer solution

To translate the LP relaxation solution from the column generation algorithm to an integer solution, a standard BP formulation is solved using a commercial solver. The formulation is equivalent to

the formulation in Section 2.1. Recall that the BP formulation of the problem is as follows.

$$\min \sum_{w\in\mathcal{W}} \sum_{i\in\mathcal{S}_w^z} C_w x_i^{w,z} + C_\nu \nu \tag{15}$$

$$\text{S.t.} \sum_{w\in\mathcal{W}} \sum_{i\in\mathcal{S}_w^z} A_{ij}^t x_i^{w,z} \geq (1-\nu) d_j^t, \qquad \forall j \in \mathcal{A}, \quad \forall t \in \mathcal{T} \tag{16}$$

$$\sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{A}} \sum_{i\in\mathcal{S}_w^z} B_{ij}^{t,w} x_i^{w,z} \geq min_w, \qquad \forall w \in \mathcal{W} \tag{17}$$

$$\sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{A}} \sum_{i\in\mathcal{S}_w^z} B_{ij}^{t,w} x_i^{w,z} \leq max_w, \qquad \forall w \in \mathcal{W} \tag{18}$$

$$x_i^{w,z} \in \{0,1\}, \qquad \forall i \in \mathcal{S}_w^z, \quad \forall w \in \mathcal{W} \tag{19}$$

For $z = 1$, additional constraints prevail:

$$\sum_{i\in\mathcal{S}_w^1} P_{ip}^w x_i^{w,z} \leq 1, \qquad \forall p \in \mathcal{S}_w^0 \tag{20}$$

$$\sum_{w\in\mathcal{W}} \sum_{p\in\mathcal{S}_w^0} \sum_{i\in\mathcal{S}_w^1} P_{ip}^w x_i^{w,z} \geq \gamma |P|, \tag{21}$$

As the formulation is already introduced earlier in this thesis, the full clarification of the constraints is not repeated in this section.

There are several reasons why the above formulation is implemented. Faster methods to solve a BP model exist in the literature, such as relax and fix algorithms, but this thesis has two simultaneous objectives: speed and solution value. Faster algorithms often provide worse solutions in comparison to the BP formulation. To be beneficial over the benchmark method, the solution needs to be better (or at least as good) while remaining within the benchmark time restriction. In this case, methods such as relax-and-fix lose their competitive edge and hence the BP formulation is implemented. Moreover, a relax-and-fix method might end up in a local optimum. Although this local optimum can be close to the global optimum, no guarantees exist. As the margins of improvement for the model in comparison to the benchmark method are small, this could result in not improving on the benchmark method. Therefore, the BP formulation above is implemented.

5 Data description

Data used in this thesis follows from practice. The data describes the number of teams required at a certain filter during an interval. The information regarding the number of teams during a period is specified in thirty-minute intervals. For scheduling reasons, the data is translated to five-minute intervals. This implies there are a total of 288 intervals for any given day. A small example showing the structure of the data can be found in Table 1. The example shows the demand for a few intervals for a four-filter instance.

Table 1: Example demand data for four-filter instance

Time		Filters			
Interval	Real start time interval	1	2	3	4
⋮	⋮	⋮	⋮	⋮	⋮
207	17:15	3	3	5	8
208	17:20	3	3	5	8
209	17:30	3	3	5	8
210	17:35	5	3	7	8
211	17:40	5	3	7	8
⋮	⋮	⋮	⋮	⋮	⋮

On the left-hand side of the table, one can find the interval number and the start time of the interval. The intervals 207 to 211 are shown, which start between 17:15 and 17:40. On the right-hand side, for every filter i, $i = \{1, 2, 3, 4\}$, one can find the number of teams demanded during these intervals. For instance, for interval 207 with start time 17:15, 3 teams are demanded at filter 1, 3 teams are demanded at filter 2, 5 teams are demanded at filter 3 and 8 filters are demanded at filter 4. Every filter has its own demand pattern. To illustrate the differences, consider Figure 3. Figure 3 is based on the demand for security teams for a Friday and Saturday.

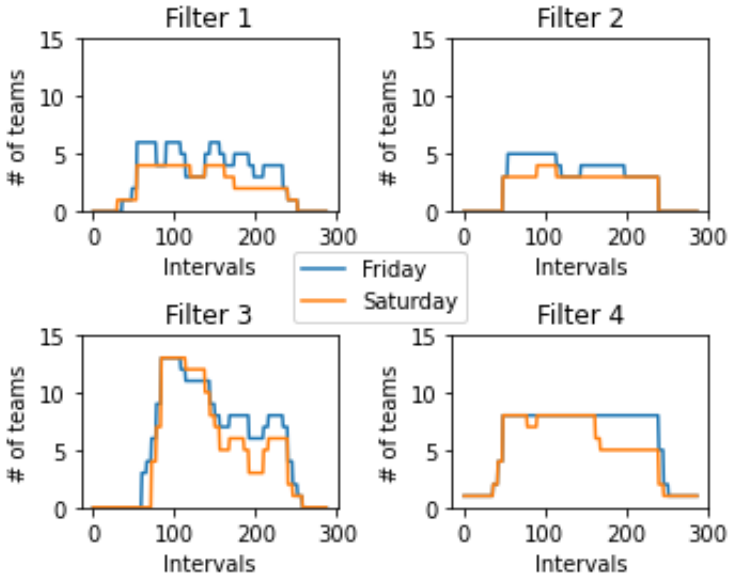

Figure 3: Example of difference in demand per day and per filter

In the above visualisation of demand, one can observe that demand fluctuates substantially by time. Moreover, different filters have different demand patterns and the height of the peak demands differ to a large extent. Likewise, demand varies considerably by day. Although this is just one example, similar differences can be observed using other data.

5.1 Fixed parameter values

Some parameter values in this thesis are fixed, as they follow from either the collective labour agreement or the labour laws specific for the Netherlands. In this section, the values for these parameters are specified.

First, let us consider the regulations regarding the long break requirement. By the collective labour agreement, in case the shift is shorter than $\beta_1 = 330$ minutes (5.5 hours), there is no long break requirement. In case the shift is longer than $\beta_1 = 330$ minutes and shorter than $\beta_2 = 480$ minutes (8 hours) there is a uninterrupted break requirement of $b_1 = 30$ minutes. In case a shift is longer than $\beta_2 = 480$ minutes, there is a break requirement of $b_2 = 45$ minutes, out of which $b_1 = 30$ minutes should be uninterrupted.

There is also strict regulation regarding the placement of the long break in a shift. Depending on the shift length, the break needs to be ξ minutes away from the start and end time of the shift. In case the shift is shorter than $\alpha = 8$ hours, $\xi_1 = 120$ minutes. In case a shift is larger than $\alpha = 8$ hours, the break needs to be at least $\xi_2 = 180$ minutes away from the start and end of the shift.

The minimum standing time is equal to $T_{min} = 60$ minutes. The maximum standing time $T_{max} = 120$ minutes. Moreover, the walking time between filters is pre-measured. Recall that the walking time is the time from the start filter to the destination filter with a stop in the break room. One can find the walking times in Table 2 below.

Table 2: Walking time (minutes)

Start filter \ Destination filter	1	2	3	4
1	5	10	15	5
2	10	5	10	5
3	15	10	5	10
4	5	5	10	5

In case an instance with three filters is considered, filter number four can be excluded from the above table.

5.2 Specific parameter values

In this section, parameters that are specific to the situation in this thesis are established. First, consider the shift lengths. In total, four shift lengths exist. including subcategories, this adds up to 6 shift types for $w \in \mathcal{W}$. Shifts of type $1a$ have a total length of 3.75 hours, with a fixed starting time at $s_{1a} = 6{:}00$. Similarly, shifts of type $1b$ have a total length of 3.75 hours as well, but with a fixed starting time at $s_{1b} = 16{:}15$. Furthermore, shifts of type $2a$ have a length of 9.5 hours and a starting time in the interval [1:15, 4:30]. Shifts of type $2b$ also have a length of 9.5 hours and a

starting time in the interval [11:00, 14:15]. Lastly, shift types 3 and 4 have lengths of 6.75 and 8.5 hours, respectively. These shift types have no requirements regarding the starting times of their shifts, except that the shifts have to end the same day as they started. This rule has one exception, namely the night shift that couples the two days together. This is a shift of 8.5 and the working hours are counted on the day that the shift began. Shift types $1a$ and $1b$ need to have exactly 3 shifts each, while shift types $2a$ and $2b$ have a minimum of 2 shifts and a maximum of 3 shifts each. Lastly, there are no requirements regarding the number of shifts of type 3 and 4.

Contrary to shifts with shift length shorter than $\beta_2 = 480$ minutes, shifts with length longer than $\beta_2 = 480$ minutes get paid for 15 out of the 45 minutes of their long break. On the basis of the shift length and the break requirement, one can now determine the paid hours for shifts of type $C_w, w \in \mathcal{W}$. Shift of type $1a$ and $1b$ have cost $C_1 = 3.75$. Shifts of type $2a$ and $2b$ have cost $C_2 = 9.0$. Lastly, shifts of type 3 and 4 have costs $C_3 = 6.25$ and $C_4 = 8.0$, respectively.

Lastly, there is a requirement regarding the maximum number of short breaks. This requirement exists due to social considerations and is represented by the parameter $SB^w, w \in \mathcal{W}$. The parameters have value $1, 1, 3, 3, 1, 2$ for shifts of type $1a$, $1b$, $2a$, $2b$, 3 and 4, respectively.

An overview of the requirements can be found in Table 3.

Shift type	1a	1b	2a	2b	3	4
Length (h)	3.75	3.75	9.5	9.5	6.75	8.5
Objective length (h)	3.75	3.75	9.0	9.0	6.25	8.0
Long break requirement (m)	0	0	45 (15 paid)	45 (15 paid)	30	30
SB^w (#)	1	1	3	3	1	2
ξ (m)	0	0	180	180	120	180
Minimum (#)	3	3	2	2	0	0
Maximum (#)	3	3	3	3	∞	∞

Table 3: Shift requirement data

Regarding the initialisation process, the following settings are used. First of all, for instances with three and four filters, the initialisation algorithm is applied to one week of data. For larger instances, as they are independent of which day it is, five days of data are used. The time restriction for generating shifts for each instance in the initialisation process is set at 300 seconds. Furthermore, for component 3 the parameter ψ that is the probability of a shift not being present in the optimal solution but being included in the start shift, equals 0,10.

Lastly, regarding the similar shifts for period 1, δ is set at a thirty-minute difference in starting interval.

5.3 Stopping criteria

Two methods are considered in this thesis: a column generation with exact pricing problem and a column generation heuristic. These column generation algorithms follow the procedure described in Algorithm 1. The column generation with exact pricing algorithm uses the following stopping criteria: maxTimeLimit has a value of 250 seconds and maxWithoutImprovement has value 8. For the column generation heuristic, the maxTimeLimit also equals 250 while maxWithoutImprovement equals 18. An iteration is considered an improvement if there is a difference between the new objective value and the old objective value of at least 1e−6.

6 Experiments

To assess the quality of the models proposed in this thesis, several experiments are conducted. The first experiment concerns a comparison between the currently used practical model (benchmark model) and the column generation model, with either an exact or a heuristic algorithm as method to the pricing problem. The first experiment is discussed in Section 6.1. The results of this experiment are discussed in Section 6.2. The second experiment assesses the scalability of the column generation model and is discussed in Section 6.3. The outcomes of the scalability experiment are reviewed in Section 6.4.

All experiments in this section were performed on a DELL Inspiron 5490 with 12 GB of RAM, running an Intel Core^{TM} i7-10510U CPU clocked at 1.80 GHz, with Microsoft Windows 10 Home. The experiments are carried out using Python 3.7.9 64-bit using the Spyder environment, version 4.1.5. Moreover, the commercial solver Gurobi Optimizer, version 9.0.3, is used.

6.1 Comparison against benchmark method: experimental setup

The first experiment is a direct comparison between the model currently used in practice and the column generation models designed in this thesis. The models are compared for both information moments, thus for the demand of four weeks before ($z = 0$) and for the demand of one week before ($z = 1$).

The benchmark model solves a Binary Programming (BP) model with a large number of shifts. The shifts are modelled as variables and the model is solved using commercial solver Gurobi. As this approach produces a model with over 200,000 variables, the BP model does not solve to optimality. For this reason, a time restriction is included, which is set at 600 seconds. Additionally, the time restriction is also included for practical reasons. The quality of the solution is measured by using the discount rate (VDC). The VDC is a measure determined by dividing the total paid hours planned (TH) by the number of hours demanded (TD). In mathematical terms, this is equivalent to

$$VDC = \frac{TH}{TD} \tag{22}$$

The interpretation of the VDC is as follows: suppose the VDC equals 1.17. This implies that in comparison to the number of hours demanded by the airport, the solution needs 1.17 times the number of hours demanded when all restrictions are satisfied. As there are rules and regulations the solution has to conform to, we know that the VDC has a minimum value of 1. In addition, the standard deviation of the VDC is measured. In addition, for the comparison against the benchmark method, an additional measure is used: the weighted VDC. By using the weighted VDC we want to determine a weighted average over the hours. The weights are determined by taking the proportion of hours over that week and divide it by the total number of hours for the full planning horizon. The weighted average informs us at which times which model performs better. Furthermore, as final performance measure, we have the standard deviation of the VDC. The standard deviation of the VDC informs us about the variability of the solutions of the instances.

For the comparison, a set of instances is available. Regarding the experiment in this section, 35 instances (5 weeks) are available for both three and four filters. For both sizes, 7 of the 35 instances (1 week of data) are used to create an initial solution. The comparison between methods is made based on the remaining 28 instances. The instances weeks that are used for comparison are shown in Table 4.

Table 4: Data weeks used

Filters	Which weeks	Initialisation
3	3, 11, 17, 29 (all 2019)	13 (2019)
4	51 (2019), 1, 5, 9 (all 2020)	45 (2019)

The experiment in this section consists of two sub-experiments. First of all, a direct comparison can be made between the benchmark method and the column generation method for period 0 (when $z = 0$). In the second sub-experiment, the practical situation is considered. The practical situation works as follows. First, the method is solved for $z = 0$. From there, the final solution that follows from $z = 0$ is used as a starting point, in combination with the demand data for period 1, for period 1 ($z = 1$). The importance here is that the solutions from period 0 are used. Because the start solution transfers from the results of period 0 and translation of results between methods are difficult, the instances considered for period 1 are different. In addition, for period 1, different implementations of constraints (7) and (8) are used. The distinction is as follows. For the benchmark method, 60% of the shifts constructed in period 0 need to reappear in that exact manner in the solution for period 1. For the CG methods of this thesis, 60% of start times of the solution of period 0 has to return similarly (start time within δ intervals) in the solution of period 1. The different implementation is used as the implementation in this thesis is more flexible and future plans exist to include this formulation in the benchmark method. Since the column generation and benchmark method work in a fundamentally different way, the methods cannot be interchangeably used without translation between the results. For this reason, no crossovers between methods are considered in this thesis. Because of the above-described different starting points and constraint

differences, no direct inferences can be made about the comparison between the methods for $z = 1$. Nevertheless, period 1 is part of the practical situation and therefore the results are presented in this thesis.

For period 1, since the minimal hours demanded might differ between periods, the VDC for period 1 is determined based on the demand for period 1.

6.2 Results: comparison against the benchmark method

This section is structured as follows. First, the outcomes for the three-filter instances are discussed. Secondly, the results for the four-filter instances are reviewed. Within the discussion of each size, first the results for period 0 ($z = 0$) are discussed. Secondly, the results for period 1 ($z = 1$) are reviewed.

6.2.1 Three filters

Period 0

The data for period 0 is provided four weeks before the actual process. The summary statistics can be found in Table 5, the full result in Table 13.

Table 5: Summary statistics for three-filter instances for period 0

Week	Benchmark		CG exact pricing			CG heuristic		
	VDC	Std. Dev.	VDC	Std. Dev.	Avg. time (s)	VDC	Std. Dev.	Avg. time (s)
3	1.2341	0.0114	1.2211	0.0107	450.39	1.2186	0.0044	465.46
11	1.2338	0.0087	1.2194	0.0091	474.79	1.2196	0.0099	440.39
17	1.1953	0.0041	1.1907	0.0023	550.70	1.1858	0.0018	498.87
29	1.2027	0.0021	1.1912	0.0026	548.25	1.1869	0.0030	504.49
Avg	1.2165	0.0192	1.2056	0.0164	506.03	1.2027	0.0173	477.30
W.Avg	1.2119		1.2018			1.1984		

Table 5 informs us about the summary statistics for the three-filter instances for period 0. The week numbers can be found under the column week. In the table, one can find the average VDC and the average standard deviation of the VDC for the individual weeks. The results show that the exact and heuristic column generation methods tend to outperform the benchmark method. The CG with exact pricing performs on average 1.09% better, while the CG heuristic performs 1.38% better. One can notice that the average variability of the instances is between 1.92% for the benchmark instances and 1.64% for the exact instances. Hence, the volatility between methods is very similar. To be complete, the CG with exact pricing problem outperforms the benchmark method on 24 of the 28 instances (85.71%), while for 2 out of the 28 instances the solutions are equal (7.14%). For the remaining two instances, the exact CG has a worse solution (7.14%). The average

improvement for the CG with exact pricing over the benchmark method is 5.48 hours, while for the instances in which the benchmark method outperforms the CG with exact pricing, the average improvement is 2.63 hours. The heuristic method performs better on 27 out of the 28 instances (96.43%) and has an equal solution value for one instance (3.57%). The average improvement for the CG heuristic over the benchmark method is 6.19 hours. Secondly, the weighted average VDC of the methods shows very similar results. Although the benchmark method seems to do relatively slightly better than the other methods, the difference is very modest and nothing can be concluded from the difference. Lastly, the average standard deviation informs us about the variability among several weeks and is very comparable among methods.

In addition to the improved results, the column generation methods tend to be faster. The benchmark method has a maximum computational time of 600 seconds. With additional time for writing the results and initialisation, the total time for one instance comes close to 650 seconds. The exact and heuristic methods have a maximum computational time of 500 seconds. Including additional writing time, the maximum time for one instance equals around 550 seconds. More computational time tends to result in only small additional gains in terms of VDC. Regarding the results, we can conclude the column generation methods provide better results in a more timely fashion. To be precise, in comparison with the benchmark method, the exact method is about 144 seconds faster on average while for the heuristic method this equals about 173 seconds.

Although the percentages of improvement are relatively small, the economic consequences are substantial. In total, to perform all shifts, 15030 hours are needed to complete the full set of benchmark shifts. In comparison, the exact method needs only 14903.75 hours and the heuristic method needs 14863 hours. To be precise, the heuristic method needs 167 hours less to solve the instances. For every hour of work in the solution, five team members need to be present. Hence, in total, 835 hours of work need not be paid. Even when considering the minimum hourly wage of 12 euros per hour without taking into account irregularity allowance, this would add up to about 10,020 euros for the four weeks considered in this thesis. Suppose that our four-weeks would be representative of the year, this would result in yearly savings of at least 130,000 euros! In fact, the number would be even higher since pensions, clothing, additional training and many more costs are not included in this number.

Period 1

The data for period 1 is provided a few days before the schedule is operational. The staff already received some indication regarding their working times and hence in comparison with period 0, another restriction is included. A minimum of 60% of similar shifts in comparison with the first period has to be included. Two shifts are similar when the starting times of the two are within δ intervals of each other. The summary results can be found in Table 6. The full results for three filters for period 1 can be found in Table 14.

Table 6: Summary statistics for three-filter instances for period 1

Week	Benchmark		CG exact pricing			CG heuristic		
	VDC	Std. Dev.	VDC	Std. Dev.	Avg. time (s)	VDC	Std. Dev.	Avg. time (s)
3	1.2330	0.0094	1.2305	0.0091	540.28	1.2310	0.0113	528.22
11	1.2336	0.0108	1.2289	0.0066	540.14	1.2287	0.0096	531.86
17	1.1938	0.0052	1.1913	0.0042	553.47	1.1952	0.0032	553.14
29	1.1974	0.0048	1.1953	0.0038	553.98	1.1984	0.0038	557.33
Avg	1.2145	0.0205	1.2115	0.0193	546.97	1.2133	0.0184	542.64
W.Avg	1.2095		1.2067			1.2090		

Although no hard claims can be made on the basis of the results described in Table 6, the methods seem to perform very similarly on their respective instances. Relatively to the results in period 0, The benchmark method improves slightly regarding the average VDC (-0.2%), while the column generation with exact pricing problem loses 0.59% in efficiency and the CG heuristic loses 1.06% efficiency. The loss in efficiency might be due to a slightly different implementation of constraints 7 and 8 in comparison to the benchmark model or the increased difficulty of the model due to the additional constraints. For the weighted average, we see that, relative to the unweighted average, the benchmark method improves about 0.07% more than the other method. Although the difference is small, it might be an indication of the column generation methods being relatively more efficient on instances with lower demand than on instances with higher demand. Moreover, we find that all the weighted averages result in lower values than the unweighted averages. This indicates that the methods perform relatively better on instances with more demand than on instances with lower demand.

Overall, on the basis of the above results and the earlier described problems regarding the instances, it is hard to draw any conclusions regarding the results of period 1.

6.2.2 Four filters

Period 0

The results for period 0 for instances with four filters are shown in Table 7.

Table 7: Summary statistics for four-filter instances for period 0

Week	Benchmark		CG exact pricing			CG heuristic		
	VDC	Std. Dev.	VDC	Std. Dev.	Avg. time (s)	VDC	Std. Dev.	Avg. time (s)
51	1.2222	0.0103	1.2210	0.0080	548.03	1.2110	0.0103	536.15
1	1.2230	0.0064	1.2242	0.0052	547.82	1.2099	0.0046	541.20
5	1.2314	0.0102	1.2365	0.0079	546.14	1.2290	0.0149	546.83
9	1.2329	0.0098	1.2326	0.0078	547.53	1.2226	0.0076	533.60
Avg	1.2274	0.0105	1.2286	0.0096	547.38	1.2181	0.0129	539.45
W.Avg	1.2270		1.2280			1.2174		

Table 7 shows the average results for four data weeks. The full results for the instances with four filters for period 0 can be found in Table 15. Table 7 shows that the benchmark and exact method perform similarly by comparing the VDC. The heuristic method scores 0.93% better than the benchmark method. The standard deviation between the methods is very similar. Considering the results of all instances, we find that the exact method scores better on 13 of the 28 instances (46.43%) with an average improvement of 2.81 hours, while for 3 of the 28 instances (10.71%) the solution values are equal. For the remaining 12 instances (42.86%), the benchmark method scores better, with an average improvement of 4.13 hours. The CG heuristic scores better on 22 of the 28 instances (78.57%) and has an average improvement of 5.51 hours. For one instance (3.57%), the results are equal. For the remaining five instances (17.86%) the benchmark method scores better. Both the column generation algorithms are close to 100 seconds faster than the benchmark method. Lastly, when considering the weighted average over the methods, we find that the weighted averages are very similar to the averages. This is an indication that the instances considered in this thesis have relatively similar demand. Therefore, it is hard the effects of varying demand on the average results.

Period 1

The summary results for the instances with four filters for period 1 can be found in Table 8. The full results can be found in Table 16.

Table 8: Summary statistics for four-filter instances for period 1

Week	Benchmark		CG exact pricing			CG heuristic		
	VDC	Std. Dev.	VDC	Std. Dev.	Avg. time (s)	VDC	Std. Dev.	Avg. time (s)
51	1,2229	0,0107	1,2327	0,0107	552,09	1,2256	0,0136	566,06
1	1,2373	0,0071	1,2468	0,0081	545,37	1,2435	0,0137	561,11
5	1,2195	0,0094	1,2310	0,0134	550,84	1,2313	0,0096	565,96
9	1,2331	0,0122	1,2472	0,0073	549,01	1,2416	0,0099	560,10
Avg	1,2282	0,0122	1,2394	0,0125	549,32	1,2355	0,0137	563,31
W.Avg	1,2286		1,2396			1,2355		

As for the results for three-filter instances, the benchmark method tends to perform similarly in terms of results between period 0 and period 1. The CG with exact pricing loses 1.06% in solution value and the CG heuristic loses 1.74% in efficiency. Between the three and four-filter instances, the results tend to show similar effects when solving for period 1. The weighted averages are very similar to the weighted averages for the methods. This is due to the fact that the demands for the four weeks are very similar. Therefore, no effects can be determined on the basis of the weighted averages.

6.3 Scalability experiment: experimental setup

In the second experiment, the following question is considered. How do the methods designed in this thesis perform in comparison to each other, while keeping the time restriction of 600 seconds in place and the number of filters in the model increases? This question is relevant as Schiphol Airport has been growing steadily for years and therefore the number of filters might increase within the near future. Therefore, two experiments are considered. First of all, to answer the above research question, a comparison is made between the methods presented in this thesis for instances with filter sizes five, six, eight and ten with a time restriction of 600 seconds. With this experiment, the goal is to investigate how the methods compare with each other when larger instances are considered. Secondly, the impact of computational time on the performance of our methods for larger instances is assessed in the following experiment. For this experiment, six instances are generated. From these instances, three instances have five filters and the remaining instances consist of ten filters. Every instance has randomly generated walking times. For each of the six instances, the model runs with the time restriction equal to 10, 20, 30, 40, 50 and 60 minutes. The goal of the experiment is to get a sense of what the effect of the time restriction and the walking time has on the performance of our model.

Let us first consider the first sub-experiment. In Section 6.1, performance is measured for instances with three and four filters. In the experiment in this section, more filters are considered. To be precise, in the scalability experiment, instances with five, six, eight and ten filters are considered. There is no data available for instances of these sizes. Therefore, one needs to create data sets of

these sizes. The approach chosen to create the data is as follows. A substantial amount of data is available for three and four filters instances. From these instances, one can deduct the demand for a single filter. All the single filter data are bundled in a large set. From there on, one can create instances of any size by randomly picking a selection of filters to represent such an instance. This approach has the advantage that demand patterns are mimicked whilst not being predictable, as the variety among filter demands is substantial. Moreover, in case of larger instances, walking time is not established. As the distance between filters remains constant, a set of walking times between filters is randomly created, with walking times between either 5, 10 or 15 minutes. These numbers are in line with the numbers for instances of three and four filters as mentioned in Table 2 in Section 5.2. To reduce the impacts of randomised walking times, two sets of instances are created for every filter size. For every set of instances, the scalability experiment works as follows. First of all, twenty instances are created using the above-described method. From there, an initialisation of shifts is created by using the initialisation algorithm. This initialisation process uses five out of the twenty instances. Finally, the other fifteen instances of the set are then used to determine our measures. Therefore, in total, one hundred sixty instances are created, with instance sizes equal to five, six, eight and ten filters.

No information regarding the optimal objective values of instances is available. It might be the case that larger instances have efficiency advantages - this is suggested by the three-filter instances, where weeks with more demand (weeks 17, 29) provide better results than weeks with less demand (weeks 3,11). The same reasoning can be applied to instances with more filters. Hence, one might argue that for larger instances in a model with unrestricted time, the optimal VDC might be lower. Consequently, no immediate claims can be made between sets of instances of different sizes. Nevertheless, this experiment informs us which method performs better within the given time restrictions and filter size and how large the impact of more filters is on the solution value.

The second sub-experiment is performed to determine the effect of the time restrictions on the methods. In this sub-experiment, three instances of both five and ten filters are considered. For these instances, six time restrictions are considered. By this experiment, we want to determine what the result is of increasing the time for larger instances and the consequent effects on the solution values.

6.4 Results: scalability experiment

This section is divided into two components. In the first component, Section 6.4.1, the results of the first sub-experiment are discussed. In the second component, Section 6.4.2, the results of the second sub-experiment are reviewed.

6.4.1 Results: larger instances with fixed time-restriction

In this section, the outcomes of the scalability experiment are discussed. The objective of the scalability experiment is determining the effect on the quality of our solution when the number of

filters in the model is increased, but the time restriction remains intact. The results in this section are ordered by means of ascending size.

Within each subsection, the average outcomes are reported for both instance sets. The outcomes for all individual instances can be found in the appendix.

Five filters

Testing our methods with instances of five filters yield the following summary results. The full results can be found in Tables 17 and 18.

Table 9: Summary statistics for sets with five filters

Instance set	CG exact pricing		CG heuristic	
	VDC	Std. Deviation	VDC	Std. Deviation
5.1	1.255	0.0256	1.257	0.0326
5.2	1.2418	0.0259	1.2422	0.0323
Average	1.2485	0.0266	1.2497	0.0334

In Table 9 one can find the results for two sets with fifteen sub-instances. The numbers reported are the averages. Overall, the results for both the CG with exact pricing and the CG heuristic are very comparable. The VDC shows, on average, slight differences between the instances of about 0.001%. Of the instances of set 5.1, 53.33% has a lower solution value using the CG with exact pricing, 6.67% has an equal value and for 40% the CG heuristic provides a lower solution. On average, the CG with exact pricing improves 4.69 hours when over the CG heuristic and if the CG heuristic provides better results than the CG with exact pricing, the improvement is on average 5.58 hours. For instances in set 5.2, the division is the same. The CG with exact pricing provides an average improvement of 3.125 hours and the CG heuristic provides an average improvement of 6.5 hours. Therefore, it seems that the results of the CG with exact pricing are more likely to be better, but if this method performs better, the improvement is relatively small. On the other hand, if the CG heuristic provides better results than the CG with exact pricing, the expected improvement is slightly larger.

Six filters

The instances with six filters give the following summary results. The full results can be found in Tables 19 and 20.

Table 10: Summary statistics for sets with six filters

Instance set	CG exact pricing		CG heuristic	
	VDC	Std. Deviation	VDC	Std. Deviation
6.1	1.2484	0.0188	1.2362	0.0191
6.2	1.2859	0.0504	1.2731	0.0498
Average	1.2671	0.0424	1.2546	0.0420

From the table, one can conclude that the randomised walking time has a substantial effect on the solution as the difference in average VDC between instances is large. The same holds for the standard deviation. Furthermore, the CG heuristic method seems to perform slightly better than the CG with exact pricing, as on average the VDC is 1.25% lower. For instance set 6.1, 14 out of the 15 instances (93.33%) perform better with a CG heuristic than with the CG with exact pricing, with an average improvement of 11.1 hours. The remaining instance (6.67%) has an improvement of 2.75 hours. For the instances in set 6.2, the results are similar. For 14 out of the 15 instances the CG heuristic outperforms the CG with exact pricing, with an average improvement of 8.98 hours. The instance remaining improves 0.75 hours by using the CG with exact pricing over the CG heuristic.

Eight filters

The summary results for the instances with eight filters are shown in Table 11. The full results can be found in Tables 21 and 22.

Table 11: Summary statistics for instance sets with eight filters

Instance set	CG exact pricing		CG heuristic	
	VDC	Std. Deviation	VDC	Std. Deviation
8.1	1.3209	0.0311	1.2743	0.0271
8.2	1.3112	0.0235	1.2678	0.0204
Average	1.3160	0.0280	1.2710	0.0242

In comparison to smaller instances, there is a very clear distinction between the VDC of the two methods. The CG heuristic results in a 4.5% lower VDC than the CG with exact pricing. For both instance sets 8.1 and 8.2, the CG heuristic performs better on all instances, with average improvements of 40.7 and 43.42 hours, respectively.

Ten filters

The summary results for the instances with ten filters are shown in Table 12. The full results can be found in Tables 23 and 24.

Table 12: Summary statistics for sets with ten filters

Instance set	CG exact pricing		CG heuristic	
	VDC	Std. Deviation	VDC	Std. Deviation
10.1	1.3692	0.0231	1.2792	0.0136
10.2	1.4054	0.0638	1.2929	0.0273
Average	1.3877	0.0521	1.2862	0.0230

The results in Table 12 show large differences between the exact and heuristic methods. Moreover, the standard deviation of the heuristic method is smaller and therefore the results of the heuristic are less volatile. For both instance sets 10.1 and 10.2, the CG heuristic provides better results for all instances than the exact CG. The average improvements for instances 10.1 and 10.2 are 112.77 and 124.37 hours, respectively.

6.4.2 Results: larger instances with variable time-restrictions

For the experiments in this section, the following time scheme is used. For the column generation component, 70% of the total time is used. To solve the BP formulation, the remaining 30% is used. Hence, in case the time restriction is set at ten minutes, seven minutes are used for the column generation model and the remaining three minutes are used for the BP model.

For illustrative purposes, a three-minute model is included, which only uses the initial set of shifts and creates an integer solution from this set of initial shifts. As one can notice in Figures 4 and 5, the immediate drop in VDC informs us that we add the best shifts at the beginning of the column generation algorithm.

First, the effect of time on the five filter instances is discussed. Afterwards, the results for ten filter instances are discussed.

Five filters

In this section, the sensitivity of the methods with five filters with regard to the time restriction is discussed. The results can be found in Figure 4 shown below. The full results on which this figure is based can be found in Table 25.

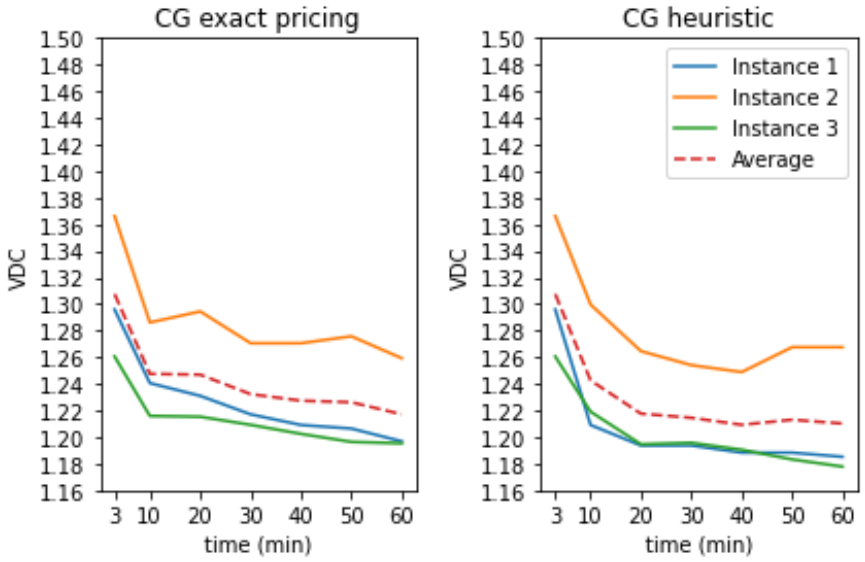

Figure 4: Effect of varying time for five filter instances

From the graph, one can notice a clear distinction between the instances. The orange line is very different in comparison to the other instances. This distinction is due to the following reasons. First of all, the orange instance has only 334.5 hours demanded (in comparison to 650.5 hours for instance 1 and 1337.5 hours for instance 3) hours demanded. In addition, the walking times of the instance are rather large. As the demanded hours are relatively low, including only one more shift has a relatively large impact on the VDC. In a similar manner, the walking time is relatively large which results in a low average effective working time for shifts in this instance. In turn, this results in a higher VDC.

It is surprising to see multiple examples of upward ticks in the results. For example, instance 2 with a computation time of fifty minutes has a slightly higher VDC (1.2758) than thirty and forty minutes (both 1.2706). The difference is due to a combination of limited solver time, a higher number of shifts in the model in case of a time restriction of fifty minutes and therefore an increased number of options. Due to the increase in the number of options, the model might take more time to find a similar or better solution. For the heuristic, the random component in the heuristic also might have some impact on the final solution which might result in upward ticks of the VDC.

Between the methods, one can notice that the differences are relatively small. Both methods result in very similar solutions. The heuristic method tends to provide better solutions fast but does not provide improvements in the longer run (when time is larger than twenty minutes). The VDC for the CG heuristic decreases on average from 1.217 at the twenty-minute mark to 1.21 at the sixty-minute mark. The exact method tends to take more time but steadily decreases in VDC, from 1.248 at the twenty-minute mark to 1.217 at the sixty-minute mark. The effect of the second instance is quite large - without this instance, the average VDC for the CG with exact pricing

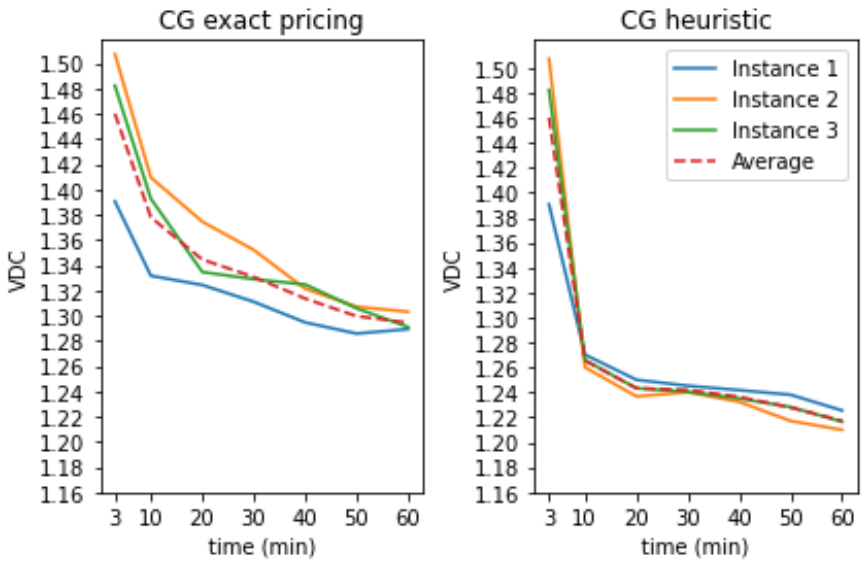

Figure 5: Effect of varying time for ten filter instances

decreases from 1.223 at the ten-minute mark to 1.196 at the sixty-minute mark, and for the CG heuristic from 1.194 at the twenty-minute mark to 1.182 at the sixty-minute mark.

Overall, the heuristic method seems to provide slightly better solutions than the exact method. The decrease for instances 1 and 3 does seem to slow down gradually as time progresses. This informs us that while additional gains can be made with regards to the VDC, these improvements are time-consuming.

Ten filters

The results of the time sensitivity analysis for ten filter instances can be found in Figure 5. The full results on which this figure is based can be found in Table 26. For the ten filter instances, the practical ten minutes show a very large decrease in the VDC in comparison with the three-minute model. The decrease for the heuristic pricing method is larger than the exact pricing method (this difference is also shown in 12). From there, the CG with exact pricing shows a larger decrease (from 1.378 at the ten-minute mark to 1.294 at the sixty-minute mark) in objective value than the CG heuristic. The heuristic method shows some decrease over time (from 1.265 at the ten-minute mark to 1.217 at the sixty-minute mark) but not as drastically as the CG with exact pricing. Similarly as with the time-sensitivity analysis for five-filter instances, we also notice a small uptick in some of the VDC values. This is due to the same reasons as in the five-filter instances.

7 Conclusions

In this thesis, we have designed a method to solve the multi-activity shift scheduling problem (MASSP) with several shift types and no fixed starting points. The main goal of this thesis was

to compare the methods designed in this thesis to the benchmark method that is used in practice. Constraints to be taken into account include restrictions on the shift types, the number of shifts of a specific type, breaks and standing times. As the algorithm is used repeatedly in practice, it should be sufficiently fast.

The main contribution of this thesis to the literature is as follows. First of all, literature regarding the MASSP is limited. In this thesis, the MASSP is combined with both several types of shifts and a large number of possible starting points, both of which are rare in the literature, especially when combined. Shift types differ based on their length, the number of breaks as well as the requirements regarding breaks. The fact that there are no fixed starting points and there are several shift types make the problem more challenging, as computation time has a big impact on the practical contribution of this thesis and both additions enlarge the number of possible shifts. Therefore, the approach in this thesis expands existing literature by providing solutions for models with increased flexibility in scheduling to create more efficient solutions to the MASSP.

The problem at hand is solved using two methods. Both methods follow a column generation approach. The first method uses an exact algorithm to solve the pricing problem. In this algorithm, pulses are iteratively sent through a network to create paths. Paths are pruned based on three pruning strategies: feasibility, objective bounds and dominance. These pruning strategies cut off large parts of the network, leaving the remaining part to be searched. As the best feasible path is never pruned, this algorithm provides the optimal path. The path is then added to the column generation formulation in case it has negative reduced costs. The second method applied in this thesis is a column generation heuristic. In this heuristic, the shifts created in the initialisation algorithm are divided into slices. These slices are then combined to create feasible shifts. The shifts with negative reduced costs are added to the column generation formulation.

The performance of the methods is tested by executing two experiments. In the first experiment, a comparison against the benchmark method for instances of three and four security filters is implemented. The methods are compared based on four measures: the average VDC, the weighted average VDC, the standard deviation of the VDC and their solving time. These measures inform us about the performance of the methods, their weighted average performance where they are weighted with the number of hours, their volatility between instances within a method and their solving time. We find that this experiment shows that for period 0 both methods constructed in this thesis yield faster and better results. The best method is the CG heuristic, which improves the VDC with on average 1.38% for the three-filter instances and 0.93% for the four-filter instances in comparison to the benchmark method. The column generation with exact method improves 1.09% for three-filter instances and performs similarly for the four-filter instances as the benchmark method, losing only 0.12%. Secondly, when comparing the weighted VDC, we find that the decrease in weighted VDC

is very similar among methods for the same number of filters and the same period, indicating that our methods perform at least as well on low-demand instances and high-demand instances as the benchmark method does. Thirdly, the standard deviation between methods for the same period is very similar. Therefore, there is no additional fluctuation in the results in comparison to the benchmark method. The CG with exact pricing is on average 144 and 100 seconds faster for the three- and four-filter instances in comparison to the benchmark method for period 0. Similarly, the CG heuristic finds results between 173 seconds (three filters) and 100 seconds (four filters) faster than the benchmark method.

For period 1, we find that the column generation methods described in this thesis perform slightly worse in comparison to their period 0 counterparts. No comparison between methods can be made based on the instances constructed in this thesis, as the start point differs per instance type and the interpretation of the shifts that have to return is slightly different. An explanation for this effect might be that, as the CG methods in this thesis provide better results for period 0, these methods might be more fitting towards the data. Subsequently, when solving for period 1, the solutions might be too much focused on period 0 and the small changes in the demand between period 0 and period 1 might result in solutions that are worse. A second explanation could be that while including more constraints, the method that creates the integer solution has increased complexity as more constraints are included in the formulation for period 1. The inclusion of these constraints results in a higher optimality gap in the process of making the solution integer. Both reasons could be a possible explanation for the worse results. Neither one of the explanations can be excluded from the possibilities as they are dependent on the instances and their starting points and creating a translation between the results at this time is time-consuming.

In the second experiment, the effects of larger instances and more computation time on the solutions of the methods are determined. The benchmark method is not taken into account here, but one would expect that the method provides worse results for larger solutions as the boundaries of the number of shifts to be included in the model are already reached. In this experiment, a comparison is made between the CG with exact pricing and the CG heuristic methods. From the experiment, one can notice that the CG heuristic achieves improved results over the CG with exact pricing when instances grow. In addition, the CG heuristic is less dependent on time. For the CG with exact pricing, the VDC increases from 1.2485 for five filters to 1.3877 for ten filters. For the CG heuristic, the differences are a lot smaller: from 1.2497 for five filters to 1.2862 for ten filters. This result is also confirmed by our variable computation time experiment. The CG heuristic tends to improve the most in the first ten minutes. From there, the CG with exact pricing, especially for the ten-filter instances, tend to decrease with larger steps than the CG heuristic. Moreover, the variable-time experiment shows that when the number of filters is increased, the average VDC tends to decrease for a longer period.

Overall, the results of the two experiments show that the CG heuristic tends to outperform the benchmark and the CG with exact pricing on instances of three and four filters. In addition, the CG heuristic is less time-dependent and therefore can more easily provide relatively good solutions to larger instances than the CG with exacting pricing.

8 Future research

The methods in this thesis provide interesting results. For future research, several interesting improvements can be looked into. First of all, it would be of interest to conclude how the CG methods perform in comparison to the benchmark method when exactly the same constraints for period 1 are considered. This requires a translation between the results of the methods since then the methods can then be compared using the same instances. This would provide interesting results on how the methods perform on the situation in practice.

8.1 Legislation

A lot of legislation is provided in this thesis. Much of the legislation follows specifically from Dutch law and the collective labour agreement. The rules and regulations could be very different in other countries and therefore it would be interesting to see what the effect of a different set of regulations on the quality of the solutions is.

8.2 Data

The data, although randomly decided upon, has some clear impact on the results. As data is randomly made available to the author, there was no selection process to get a diverse selection of weeks over the year. The effect of this is clear. For example, in the three-filter instances, there is a clear distinction in the number of hours between the four instances: two instances need a relatively low number of hours, and two numbers need a relatively high number of hours. For the four-filter instances, the demanded hours are quite similar for all four weeks. The effect of these changes in data might impact the outcomes of this thesis. Therefore, it would be interesting to perform the same study in which random data over the year is available.

In addition, the data for four weeks is only available for winter weeks due to the fact that only at that time, four filters were used. This is due to construction work at the filters. This impacts the demand severely and therefore might have a big impact on the results. Since this is a temporary situation, it would be hard to get data over a longer period to make a more fear comparison. If it would be possible in future research, this might give interesting results.

8.3 Heuristic

The heuristic implemented in this thesis is rather simplistic. One of the consequences of such a heuristic is that reduced cost can be determined only after a full shift is created. It is not known what the reduced cost of this shift is and whether the shift has negative reduced costs. It might take a while to find a shift that has negative reduced costs, since the determination of the slices of the path is randomised. A different approach to implement a heuristic could be as follows. First of all, the starting time is determined upfront. Simultaneously, one can determine the profile. Then, for every entry in the profile, one can determine the best option to include, depending on the length and the earlier components. In this way, the effect of randomly determining all components in a shift is reduced.

In addition, the parameters of the heuristic can be further optimised. The set of parameters used in this thesis is equal for both the three- and four-filter instances for period 0 and 1 and for the larger instances of five, six, eight and ten filters. However, there might be benefits in changing the parameter set between sizes and/or periods.

9 Bibliography

Abbink, E. J., Albino, L., Dollevoet, T., Huisman, D., Roussado, J. and Saldanha, R. L.: 2011, Solving large scale crew scheduling problems in practice, *Public Transport* **3**(2), 149–164.

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

Al-Yakoob, S. M. and Sherali, H.: 2007, Mixed-integer programming models for an employee scheduling problem with multiple shifts and work locations, *Annals of Operations Research* **155**(1), 119–142.

Al-Yakoob, S. M. and Sherali, H.: 2008, A column generation approach for an employee scheduling problem with multiple shifts and work locations, *Journal of the Operational Research Society* **59**(1), 34–43.
URL: *https://doi.org/10.1057/palgrave.jors.2602294*

Borndörfer, R., Schelten, U., Schlechte, T. and Weider, S.: 2006, A Column Generation Approach to Airline Crew Scheduling, *Operations Research Proceedings 2005* (June 2014), 343–348.

Brunner, J. O. and Edenharter, G. M.: 2011, Long term staff scheduling of physicians with different experience levels in hospitals using column generation, *Health Care Management Science* **14**(2), 189–202.

CBS: 2017, How many passengers travel through Dutch airports?
URL: *https://www.cbs.nl/en-gb/faq/luchtvaart/how-many-passengers-travel-through-dutch-airports-*

CBS: 2020, Aviation; monthly figures of Dutch airports.
URL: *https://www.cbs.nl/en-gb/figures/detail/37478eng*

Côté, M. C., Gendron, B. and Rousseau, L. M.: 2013, Grammar-based column generation for personalized multi-activity shift scheduling, *INFORMS Journal on Computing* **25**(3), 461–474.

Dahmen, S., Rekik, M. and Soumis, F.: 2018, An implicit model for multi-activity shift scheduling problems, *Journal of Scheduling* **21**(3), 285–304.

Dantzig, G. B.: 1954, Letter to the Editor—A Comment on Edie's "Traffic Delays at Toll Booths", *Journal of the Operations Research Society of America* **2**(3), 339–341.

Desrochers, M. and Soumis, F.: 1989, A Column Generation Approach to the Urban Transit Crew Scheduling Problem, *Transportation Science* **23**(1), 1–13.

Edie, L. C.: 1954, Traffic Delays at Toll Booths, *Journal of the Operations Research Society of America* **2**(2), 107–138.

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

Gérard, M., Clautiaux, F. and Sadykov, R.: 2016, Column generation based approaches for a tour scheduling problem with a multi-skill heterogeneous workforce, *European Journal of Operational Research* **252**(3), 1019–1030.

Hanafi, R. and Kozan, E.: 2014, A hybrid constructive heuristic and simulated annealing for railway crew scheduling, *Computers and Industrial Engineering* **70**(1), 11–19.
URL: *http://dx.doi.org/10.1016/j.cie.2014.01.002*

Heil, J., Hoffmann, K. and Buscher, U.: 2020, Railway crew scheduling: Models, methods and applications, *European Journal of Operational Research* **283**(2), 405–425.
URL: *https://doi.org/10.1016/j.ejor.2019.06.016*

Kasirzadeh, A., Saddoune, M. and Soumis, F.: 2017, Airline crew scheduling: models, algorithms, and data sets, *EURO Journal on Transportation and Logistics* **6**(2), 111–137.

Lourenço, H. R., Paixão, J. P. and Portugal, R.: 2001, Multiobjective metaheuristics for the bus-driver scheduling problem, *Transportation Science* **35**(3), 331–343.

Lozano, L. and Medaglia, A. L.: 2013, On an exact method for the constrained shortest path problem, *Computers and Operations Research* **40**(1), 378–384.

Maenhout, B. and Vanhoucke, M.: 2007, An electromagnetic meta-heuristic for the nurse scheduling problem, *Journal of Heuristics* **13**(4), 359–385.

Pan, S., Akplogan, M., Touati, N., Lucas, L. and Calvo, R. W.: 2016, Solving a Multi-Activity Shift Scheduling Problem with a Tabu Search Heuristic, pp. 317–326.

Qu, Y. and Curtois, T.: 2020, Solving the Multi-activity Shift Scheduling Problem using Variable Neighbourhood Search, *ICORES 2020 - Proceedings of the 9th International Conference on Operations Research and Enterprise Systems* pp. 227–232.

Quimper, C. G. and Rousseau, L. M.: 2010, A large neighbourhood search approach to the multi-activity shift scheduling problem, *Journal of Heuristics* **16**(3), 373–392.
URL: *https://doi.org/10.1007/s10732-009-9106-6*

Restrepo, M. I., Gendron, B. and Rousseau, L. M.: 2018, Combining Benders decomposition and column generation for multi-activity tour scheduling, *Computers and Operations Research* **93**, 151–165.
URL: *https://doi.org/10.1016/j.cor.2018.01.014*

Restrepo, M. I., Lozano, L. and Medaglia, A. L.: 2012, Constrained network-based column gener-

ation for the multi-activity shift scheduling problem, *International Journal of Production Economics* **140**(1), 466–472.

Salvagnin, D. and Walsh, T.: 2012, A hybrid MIP/CP approach for multi-activity shift scheduling, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **7514 LNCS**, 633–646.

Soukour, A. A., Devendeville, L., Lucet, C. and Moukrim, A.: 2012, Staff scheduling in airport security service, *IFAC Proceedings Volumes (IFAC-PapersOnline)*, Vol. 14, IFAC, pp. 1413–1418.
URL: *http://dx.doi.org/10.3182/20120523-3-RO-2023.00169*

Soukour, A. A., Devendeville, L., Lucet, C. and Moukrim, A.: 2013, A Memetic Algorithm for staff scheduling problem in airport security service, *Expert Systems with Applications* **40**(18), 7504–7512.
URL: *http://dx.doi.org/10.1016/j.eswa.2013.06.073*

10 Appendix

In this appendix the full results of the instances can be found. The instances have been numbered, according to the following method. Consider the following two instance numbers: '3.11.MA' and '5.1.6'. The former is used in case the data comes from a specific day within a specific week. The interpretation is as follows. The instance consists of 3 filters, comes from week 11 and it represents demand from a Monday. The latter is used when the data is randomly created and therefore does not belong to a specific week. In a similar manner, '5.1.6' represents an instance with 5 filters, from instance set 1 and is instance 6 from this set.

10.1 Results: comparison against the benchmark method

In the results in this section one can find demanded hours, the benchmark results and VDC and the result, VDC and time for the exact CG and the CG Heuristic. FR stands for Final Result.

Three filters: period 0

Table 13: Results three-filter instances period 0

Number	Demand (h)	Benchmark		CG exact pricing			CG heuristic		
		FR (h)	VDC	FR (h)	VDC	Time (s)	FR (h)	VDC	Time (s)
3.3.MO	335	412.5	1.2313	404.5	1.2075	533.21	407.25	1.2157	424.82
3.3.TU	282	351	1.2447	344.75	1.2225	298.34	344.75	1.2225	447.19
3.3.WE	303	380.5	1.2558	371.5	1.2261	416.83	369.75	1.2203	461.95
3.3.TH	349	431	1.2350	423.75	1.2142	421.37	424.75	1.217	525.93
3.3.FR	346	423	1.2225	423	1.2225	501.53	421.25	1.2175	421.2
3.3.SA	271	332.25	1.2260	336.75	1.2426	544.69	332.25	1.226	511.82
3.3.SU	381	466	1.2231	462	1.2126	436.78	461.5	1.2113	465.31
3.11.MO	347	427.25	1.2313	419.25	1.2082	408.9	423.75	1.2212	442.02
3.11.TU	303	379.25	1.2517	372.25	1.2285	459.1	370.25	1.2219	442.59
3.11.WE	305.5	378.5	1.2390	375.75	1.23	542.57	376	1.2308	398.1
3.11.TH	346.5	425.75	1.2287	420.25	1.2128	472.36	419.25	1.21	422.92
3.11.FR	356	436.25	1.2254	434.5	1.2205	449.75	431.75	1.2128	482.52
3.11.SA	280.5	346.5	1.2353	344.5	1.2282	550.07	346.25	1.2344	427.28
3.11.SU	373	457	1.2252	450.5	1.2078	440.78	449.75	1.2058	467.29
3.17.MO	567	676.75	1.1935	672.5	1.1861	548.54	673.25	1.1874	455.2
3.17.TU	558.5	669.75	1.1991	665.25	1.1911	547.96	663.25	1.1876	545.91
3.17.WE	549	657.25	1.1971	653.75	1.1908	552.89	649.75	1.1835	460.94
3.17.TH	560	672.25	1.2004	668.75	1.1942	553.49	664.25	1.1862	457.67
3.17.FR	577	686.75	1.1902	686.75	1.1902	554.74	685.25	1.1876	546.32
3.17.SA	556.5	661.75	1.1891	662.5	1.1905	550.75	659.75	1.1855	546.07
3.17.SU	595.5	713.25	1.1977	710	1.1923	546.51	704.5	1.183	479.96
3.29.MO	573.5	689.25	1.2018	682.25	1.1896	548.21	681.25	1.1879	546.64
3.29.TU	542	650.25	1.1997	646.5	1.1928	550.24	641.5	1.1836	520.69
3.29.WE	538.5	647.5	1.2024	641	1.1903	547.99	638.5	1.1857	520.6
3.29.TH	562	676.75	1.2042	672.25	1.1962	547.72	670.5	1.1931	458.73
3.29.FR	567	684	1.2063	675.75	1.1918	549.24	673	1.1869	482.45
3.29.SA	523	628	1.2008	622.5	1.1902	548.23	619	1.1836	456.26
3.29.SU	556.5	669.75	1.2035	660.75	1.1873	546.09	660.75	1.1873	546.09

Three filters: period 1

Table 14: Results three-filter instances period 1

Number	Demand (h)	Benchmark		CG exact pricing			CG heuristic		
		FR (h)	VDC	FR (h)	VDC	Time (s)	FR (h)	VDC	Time (s)
3.3.MO	342	423.25	1.237573	418	1.2222	541.18	417	1.2193	546.62
3.3.TU	273.5	342	1.250457	336.75	1.2313	537.64	337.5	1.234	492.92
3.3.WE	293	364.25	1.243174	360.75	1.2312	539.04	365.25	1.2466	540.02
3.3.TH	351.5	428.5	1.219061	433.75	1.234	541.1	428.5	1.2191	542.9
3.3.FR	355.5	440	1.237693	433.75	1.2201	541.65	438.25	1.2328	513.43
3.3.SA	265	324.25	1.223585	328.5	1.2396	539.94	326	1.2302	540.04
3.3.SU	373.5	457	1.223561	457	1.2236	540.45	455.25	1.2189	547.08
3.11.MO	347.5	429.25	1.235252	427.5	1.2302	542.18	429.25	1.2353	544.03
3.11.TU	296	370.5	1.251689	369.5	1.2483	538.85	369.5	1.2483	540.72
3.11.WE	312	385.75	1.236378	384	1.2308	538.58	385	1.234	477.14
3.11.TH	346	423	1.222543	423.75	1.2247	541.8	424	1.2254	545.75
3.11.FR	357.5	439.25	1.228671	437.25	1.2231	540.61	436.5	1.221	544.97
3.11.SA	283.5	350	1.234568	350.75	1.2372	538.66	351.75	1.2407	542.58
3.11.SU	374	457	1.221925	456	1.2193	541.3	453.5	1.2126	502.36
3.17.MO	567.5	675	1.189427	674.25	1.1881	553.47	677.75	1.1943	553.13
3.17.TU	559	671.5	1.201252	666	1.1914	552.27	669	1.1968	553.9
3.17.WE	549.5	660	1.201092	652.75	1.1879	552.17	659	1.1993	553.83
3.17.TH	560.5	669.75	1.194915	669.75	1.1949	556	672.25	1.1994	554.04
3.17.FR	577.5	687.5	1.190476	685.75	1.1874	553.87	687.75	1.1909	540.69
3.17.SA	557	664.5	1.192998	662.5	1.1894	552.49	663.75	1.1917	556.02
3.17.SU	596	707.25	1.186661	715	1.1997	554.02	711.5	1.1938	560.39
3.29.MO	567	681.25	1.201499	679	1.1975	555.94	677	1.194	556.41
3.29.TU	543.5	652	1.199632	647.5	1.1914	552.27	648.5	1.1932	554.91
3.29.WE	539	642	1.191095	642.25	1.1916	554.94	645.75	1.1981	556.33
3.29.TH	555	662.5	1.193694	667	1.2018	555.8	665.5	1.1991	560.37
3.29.FR	566.5	675.25	1.191968	675	1.1915	554.93	679	1.1986	560.11
3.29.SA	516	621.75	1.204942	618.25	1.1982	551.88	619.25	1.2001	555.41
3.29.SU	547.5	656.5	1.199087	654.5	1.1954	552.1	660	1.2055	557.78

Four filters: period 0

Table 15: Results four-filter instances period 0

Number	Demand (h)	Benchmark		CG exact pricing			CG heuristic		
		FR (h)	VDC	FR (h)	VDC	Time (s)	FR (h)	VDC	Time (s)
4.51.MO	386	473	1.225389	473	1.2254	547	468.75	1.2144	530.47
4.51.TU	353.5	435.5	1.231966	434.75	1.2298	548.49	432.75	1.2242	516.66
4.51.WE	372.5	462	1.240268	456.75	1.2262	548.18	455.25	1.2221	531.08
4.51.TH	492	599.25	1.217988	595.75	1.2109	547.87	592.25	1.2038	552.62
4.51.FR	527	642	1.218216	636.75	1.2083	548.09	631.5	1.1983	530.18
4.51.SA	464	560.25	1.207435	570	1.2284	547.94	564.75	1.2171	551.81
4.51.SU	515.5	626	1.214355	627.75	1.2177	548.64	617.25	1.1974	540.25
4.1.MO	461	562	1.219089	561.25	1.2175	551.25	556.5	1.2072	546.77
4.1.TU	400.5	495	1.235955	493	1.231	546.77	487.75	1.2179	514.27
4.1.WE	449.5	550.25	1.224138	550.25	1.2241	546.95	544	1.2102	549.24
4.1 TH	457	554.5	1.213348	556.5	1.2177	548.1	554.75	1.2139	549.58
4.1.FR	453	554.75	1.224614	554	1.223	547.23	547.75	1.2092	551.91
4.1.SA	398	486.5	1.222362	487.25	1.2242	547.53	481	1.2085	550.16
4.1.SU	449	548.5	1.221604	553	1.2316	546.92	539.75	1.2021	526.48
4.5.MO	380	461.5	1.214474	467.75	1.2309	545.02	460.75	1.2125	549.15
4.5.TU	313.5	389.5	1.242424	392.25	1.2512	544.91	391.25	1.248	550.58
4.5.WE	330	409.75	1.241667	408.75	1.2386	546.61	409.75	1.2417	550.36
4.5.TH	372.5	460.5	1.236242	458.75	1.2315	546.03	450.75	1.2101	523.68
4.5.FR	384	471.25	1.227214	471.25	1.2272	546.52	473	1.2318	551.49
4.5.SA	296.5	367	1.237774	368.75	1.2437	546.09	368.75	1.2437	551.07
4.5.SU	412	502.5	1.21966	507.75	1.2324	547.77	500.75	1.2154	551.5
4.9.MO	442.5	539.5	1.219209	545.25	1.2322	548.19	537	1.2136	541.84
4.9.TU	377	461.5	1.224138	467	1.2387	547.22	459.75	1.2195	520.43
4.9.WE	372	461.25	1.239919	458.75	1.2332	545.69	457	1.2285	532.43
4.9.TH	405.5	504	1.24291	497.75	1.2275	546.25	494.25	1.2189	487.89
4.9.FR	403.5	495.25	1.227385	498.75	1.2361	548.03	491.5	1.2181	550.79
4.9.SA	347.5	433.5	1.247482	432	1.2432	550.35	430.25	1.2381	552.07
4.9.SU	436	536	1.229358	530.75	1.2173	546.98	532.5	1.2213	549.73

Four filters: period 1

Table 16: Results four-filter instances period 1

Number	Demand (h)	Benchmark		CG exact pricing			CG heuristic		
		FR (h)	VDC	FR (h)	VDC	Time (s)	FR (h)	VDC	Time (s)
4.51.MO	383	467,75	1,221279	476	1,2428	546,39	475,75	1,2422	560,05
4.51.TU	349	432,25	1,238539	434	1,2436	545	433	1,2407	558,29
4.51.WE	380	469,25	1,234868	473,75	1,2467	544,09	469,5	1,2355	558,64
4.51.TH	481	584,25	1,214657	586	1,2183	555,21	589,75	1,2261	570,25
4.51.FR	513	626	1,220273	628,5	1,2251	557,21	634,25	1,2364	573,24
4.51.SA	454	548,5	1,20815	555,75	1,2241	553	549	1,2093	565,27
4.51.SU	507	619	1,220907	627,75	1,2382	558	611,25	1,2056	570,64
4.1.MO	457,5	556,75	1,21694	557,75	1,2191	550,05	561,25	1,2268	565,89
4.1.TH	385,5	477,25	1,238003	481,75	1,2497	549,85	481,75	1,2497	564,58
4.1.WE	442,5	542,25	1,225424	550,25	1,2435	554,99	546	1,2339	568,74
4.1.TH	462	559	1,209957	571,5	1,237	555,36	571,75	1,2376	567,7
4.1.FR	458	559,25	1,22107	564,75	1,2331	552,59	562,25	1,2276	569,91
4.1.SA	402	485,5	1,207711	485,5	1,2077	545,7	489,25	1,217	561,75
4.1.SU	448,5	546	1,217391	550,25	1,2269	547,32	550	1,2263	563,13
4.5.MO	353	436,5	1,236544	439	1,2436	546,77	437,5	1,2394	565,53
4.5.TU	291	362	1,243986	362,75	1,2466	542,01	365,5	1,256	561,8
4.5.WE	309,5	385,75	1,246365	387,5	1,252	545,24	385,75	1,2464	554,95
4.5.TH	351	432,75	1,232906	432	1,2308	544,57	431,25	1,2286	561,31
4.5.FR	358,5	442	1,232915	448	1,2497	546,91	440,25	1,228	562,17
4.5.SA	286,5	356,25	1,243455	360,75	1,2592	541,88	363,5	1,2688	557,89
4.5.SU	394	482,75	1,225254	490,75	1,2456	550,21	487,5	1,2373	564,13
4.9.MO	407	498,5	1,224816	501,5	1,2322	552,85	502,5	1,2346	564,17
4.9.TU	341	418,75	1,228006	426,75	1,2515	546,53	421,5	1,2361	559,33
4.9.WE	345,5	431	1,247467	432	1,2504	546,99	434,75	1,2583	554,74
4.9.TH	383,5	473,5	1,234681	480,75	1,2536	549,63	476,5	1,2425	557,69
4.9.FR	382,5	466,75	1,220261	476,5	1,2458	551,32	474	1,2392	562,37
4.9.SA	335	420,25	1,254478	420,25	1,2545	545,11	419,75	1,253	559,81
4.9.SU	413	504,75	1,222155	513	1,2421	550,62	507	1,2276	562,62

10.2 Results: larger instances with fixed time-restriction

Table 17: Results five filters instance for set 1

Number	Demanded (h)	CG exact pricing			CG heuristic		
		Result (h)	VDC	Time (s)	Result (h)	VDC	Time (s)
5.1.6	448	575.25	1.284	614.72	578.75	1.2919	615.57
5.1.7	500	628.25	1.2565	614.8	635.5	1.271	615.99
5.1.8	429.5	542.5	1.2631	614.85	545.5	1.2701	615.8
5.1.9	805.5	993.25	1.2331	615.36	998.5	1.2396	615.98
5.1.10	650	812	1.2492	614.54	812.75	1.2504	615.74
5.1.11	342	451	1.3187	615.61	458	1.3392	615.96
5.1.12	486.5	616.75	1.2677	615.46	615.75	1.2657	615.87
5.1.13	553	697.75	1.2618	614.73	705.75	1.2762	615.69
5.1.14	1013	1229.25	1.2135	615.71	1208.25	1.1927	616.06
5.1.15	861	1060.75	1.232	615.31	1057.5	1.2282	616.11
5.1.16	813.5	991	1.2182	615.19	990.25	1.2173	616.09
5.1.17	543	685	1.2615	615.3	681	1.2541	615.84
5.1.18	736	920.25	1.2503	615.05	923	1.2541	615.9
5.1.19	768	954	1.2422	615.94	954	1.2422	615.91
5.1.20	463	589.75	1.2738	615.22	586.25	1.2662	615.51

Table 18: Results five filters instance for set 2

Number	Demanded (h)	CG exact pricing			CG heuristic		
		Result (h)	VDC	Time (s)	Result (h)	VDC	Time (s)
5.2.6	892	1083	1.2141	614.62	1083.75	1.215	615.37
5.2.7	521	646	1.2399	614.32	649.5	1.2466	614.65
5.2.8	781	952.5	1.2196	614.35	937.5	1.2004	614.83
5.2.9	501	630.75	1.259	614.45	630.75	1.259	615.16
5.2.10	933.5	1140.5	1.2217	614.29	1130	1.2105	614.87
5.2.11	631.5	770.5	1.2201	614.2	767.25	1.215	615.02
5.2.12	523	658.5	1.2591	614.63	663	1.2677	615.05
5.2.13	449.5	563	1.2525	614.33	566.25	1.2597	614.76
5.2.14	520.5	645	1.2392	614.56	642.75	1.2349	614.94
5.2.15	402.5	507.75	1.2615	614.68	509.5	1.2658	614.6
5.2.16	404	527.75	1.3063	614.13	532.25	1.3175	614.73
5.2.17	623	771.75	1.2388	614.26	770	1.236	614.68
5.2.18	775.5	943.25	1.2163	614.3	944.75	1.2182	614.93
5.2.19	429.5	546.25	1.2718	614.45	551.5	1.2841	614.71
5.2.20	1136.5	1373.25	1.2083	614.78	1367	1.2028	615.06

Table 19: Results six filters instance for set 1

Number	Demanded (h)	CG exact pricing			CG heuristic		
		Result (h)	VDC	Time (s)	Result (h)	VDC	Time (s)
6.1.6	799.5	998.75	1.2492	616.83	989.25	1.2373	617.37
6.1.7	989.5	1205	1.2178	616.21	1200	1.2127	617.67
6.1.8	835.5	1056.5	1.2645	616.49	1038.75	1.2433	617.27
6.1.9	775	961	1.24	616.92	953.75	1.2306	617.6
6.1.10	839	1041	1.2408	616.76	1034	1.2324	617.16
6.1.11	481	617.5	1.2838	617.56	614	1.2765	617.96
6.1.12	531	670.75	1.2632	616.22	673.5	1.2684	617.38
6.1.13	1081	1329.25	1.2296	616.25	1302.25	1.2047	617.26
6.1.14	729	927.25	1.2719	615.9	910.5	1.249	617.32
6.1.15	746	929	1.2453	616.42	924.5	1.2393	618.39
6.1.16	750.5	937	1.2485	616.43	934.25	1.2448	617.75
6.1.17	766.5	962.5	1.2557	616.24	954.25	1.2449	617.69
6.1.18	825.5	1009.5	1.2229	616.32	1007.75	1.2208	617.51
6.1.19	774	981.25	1.2678	615.65	945.75	1.2219	617.15
6.1.20	1056.5	1294	1.2248	616.18	1285	1.2163	617.68

Table 20: Results six filters instance for set 2

Number	Demanded (h)	CG exact pricing			CG heuristic		
		Result (h)	VDC	Time (s)	Result (h)	VDC	Time (s)
6.2.6	660	849	1.2864	616.26	840.75	1.2739	617.22
6.2.7	654.5	838.25	1.2807	616.43	830.5	1.2689	617.39
6.2.8	312	450.25	1.4431	615.74	446.75	1.4319	616.89
6.2.9	859.5	1083.25	1.2603	616.43	1069.25	1.244	617.49
6.2.10	826.5	1016.75	1.2302	617.14	1006	1.2172	616.78
6.2.11	727.5	911.75	1.2533	616.24	902	1.2399	617.08
6.2.12	621.5	789.25	1.2699	616.66	785	1.2631	616.98
6.2.13	610	779	1.277	616.85	770	1.2623	617
6.2.14	698	883	1.265	617.23	876.75	1.2561	617.08
6.2.15	521.5	674	1.2924	616.86	667.75	1.2804	616.93
6.2.16	463.5	612.25	1.3209	618.06	606.75	1.3091	616.78
6.2.17	514	668.25	1.3001	616.85	655.75	1.2758	617.18
6.2.18	466	618.5	1.3273	616.69	607.75	1.3042	616.85
6.2.19	705	886.75	1.2578	615.86	887.5	1.2589	617.05
6.2.20	1228	1503.75	1.2246	616.51	1486.5	1.2105	617.32

Table 21: Results eight filters instance for set 1

Number	Demanded (h)	CG exact pricing			CG heuristic		
		Result (h)	VDC	Time (s)	Result (h)	VDC	Time (s)
8.1.6	978.5	1279.5	1.3076	622.8	1229	1.256	623.68
8.1.7	885	1165	1.3164	623.16	1132.75	1.2799	623.39
8.1.8	843.5	1092	1.2946	622.99	1060.25	1.257	624.11
8.1.9	566	765.25	1.352	622.82	742	1.311	623.96
8.1.10	1573	1999.25	1.271	623.49	1930.25	1.2271	625.02
8.1.11	1086	1413.25	1.3013	623.2	1364.25	1.2562	623.96
8.1.12	996	1317.75	1.323	622.86	1251.75	1.2568	623.83
8.1.13	1047.5	1348.25	1.2871	623.21	1307	1.2477	624.2
8.1.14	831	1091.5	1.3135	622.88	1047.25	1.2602	623.8
8.1.15	766.5	993.5	1.2962	623.43	985.5	1.2857	623.87
8.1.16	537	731.75	1.3627	622.7	712.75	1.3273	623.82
8.1.17	658	895.25	1.3606	622.99	850.75	1.2929	623.53
8.1.18	702.5	969.25	1.3797	622.57	907	1.2911	623.4
8.1.19	590.5	795.75	1.3476	622.83	773.5	1.3099	623.74
8.1.20	1064	1382.75	1.2996	623.13	1335.5	1.2552	623.87

Table 22: Results eight filters instance for set 2

Number	Demanded (h)	CG exact pricing			CG heuristic		
		Result (h)	VDC	Time (s)	Result (h)	VDC	Time (s)
8.2.6	678	901.25	1.3293	623.56	875	1.2906	623.24
8.2.7	1145.5	1512	1.3199	624.4	1430.25	1.2486	624.19
8.2.8	947	1250.25	1.3202	623.28	1202.75	1.2701	624.45
8.2.9	1590	2043.5	1.2852	623.65	1981	1.2459	624.97
8.2.10	1023.5	1327.25	1.2968	623.75	1294.5	1.2648	623.36
8.2.11	910	1210	1.3297	623.01	1171.5	1.2874	624.1
8.2.12	1133	1462	1.2904	622.96	1446.75	1.2769	624.54
8.2.13	848.5	1123	1.3235	623.13	1091.75	1.2867	624.39
8.2.14	658	902	1.3708	622.22	861	1.3085	623.31
8.2.15	743	972.25	1.3085	622.8	949.75	1.2783	623.96
8.2.16	1175.5	1502	1.2778	623.16	1464.25	1.2456	624.18
8.2.17	1315.5	1711.5	1.301	623.32	1628	1.2376	624.26
8.2.18	1042	1356.75	1.3021	623.59	1296.75	1.2445	624.06
8.2.19	746	992	1.3298	622.94	953.75	1.2785	624.04
8.2.20	1058	1357.5	1.2831	622.9	1325	1.2524	624.5

Table 23: Results ten filters instance for set 1

Number	Demanded (h)	CG exact pricing			CG heuristic		
		Result (h)	VDC	Time (s)	Result (h)	VDC	Time (s)
10.1.6	1667	2271.75	1.3628	632.05	2096	1.2573	634.31
10.1.7	1507	2040.5	1.354	632.86	1908.75	1.2666	634.34
10.1.8	923	1256	1.3608	632.6	1189.5	1.2887	632.96
10.1.9	1225.5	1681.25	1.3719	632.97	1577.25	1.287	634.04
10.1.10	1021.5	1384.25	1.3551	631.21	1298	1.2707	632.89
10.1.11	1503.5	2040.25	1.357	633.4	1921.5	1.278	634.36
10.1.12	933.5	1292.25	1.3843	632.99	1216.25	1.3029	634.48
10.1.13	1131	1511.25	1.3362	632.54	1445.5	1.2781	633.16
10.1.14	979.5	1373.25	1.402	631.81	1267.5	1.294	634.07
10.1.15	1308	1844.25	1.41	632.62	1664.25	1.2724	632.84
10.1.16	1308	1828.25	1.3977	633.35	1650	1.2615	633.56
10.1.17	1341.5	1864	1.3895	633.72	1719	1.2814	633.67
10.1.18	1308.5	1797.75	1.3739	632.99	1650.25	1.2612	634.01
10.1.19	1129.5	1497.5	1.3258	632.86	1461.75	1.2942	633.41
10.1.20	1165	1581.5	1.3575	632.14	1507	1.2936	634.48

Table 24: Results ten filters instance for set 2

Number	Demanded (h)	CG exact pricing			CG heuristic		
		Result (h)	VDC	Time (s)	Result (h)	VDC	Time (s)
10.2.6	1058	1442.5	1.3634	632.49	1357.25	1.2828	633.14
10.2.7	1372.5	1893.25	1.3794	631.82	1740.75	1.2683	632.9
10.2.8	1451	1954.75	1.3472	632.71	1832.75	1.2631	632.35
10.2.9	1093.5	1476.5	1.3503	632.23	1432.75	1.3102	633.15
10.2.10	748.5	1070.5	1.4302	632.5	1020.75	1.3637	633.31
10.2.11	848	1243	1.4658	631.24	1114.5	1.3143	633.21
10.2.12	1283.5	1713.5	1.335	632.62	1651	1.2863	634.21
10.2.13	990	1468.25	1.4831	631.12	1289	1.302	633.14
10.2.14	1254	1979.25	1.5783	631.84	1582.5	1.262	633.22
10.2.15	989	1389	1.4044	632.27	1275.25	1.2894	633.31
10.2.16	969	1314.5	1.3566	631.82	1256	1.2962	633.66
10.2.17	906.5	1233.25	1.3605	632.51	1184.25	1.3064	634
10.2.18	844	1184.75	1.4037	631.79	1112.5	1.3181	633.02
10.2.19	1369.5	1983	1.448	631.58	1713.5	1.2512	632.1
10.2.20	850	1169.5	1.3759	631.73	1087.25	1.2791	633.11

10.3 Results: larger instances with variable time-restrictions

Note that the computational time in this experiment is fixed at 10, 20, 30, 40, 50 and 60 minutes. Additionally, the results of three-minute experiments where only the initial set of shifts is included are shown. The remainder is writing and initialisation time. The numbering in this section is as follows. First the number of filters is mentioned, then the instance, then the length of the computational time. Hence, 5.2.10 is instance two with five filters and a computational time of ten minutes.

Table 25: Results time sensitivity five filters

Number	Demanded (h)	CG exact pricing			CG heuristic		
		Result (h)	VDC	Time (s)	Result (h)	VDC	Time (s)
5.1.3[1]	650,5	843	1,2959	199,24	843	1,2959	199,24
5.1.10	650,5	807	1,2406	625,93	786,5	1,2091	619,73
5.1.20	650,5	800,75	1,231	1221,95	776,5	1,1937	1217,41
5.1.30	650,5	791,75	1,2171	1818,58	776,5	1,1937	1819,05
5.1.40	650,5	786,5	1,2091	2422,69	773	1,1883	2421,99
5.1.50	650,5	784,75	1,2064	3016,54	773	1,1883	3026,13
5.1.60	650,5	778,5	1,1968	3623,78	771	1,1852	3621,64
5.2.3[1]	334,5	457	1,3662	197,34	457	1,3662	197,34
5.2.10	334,5	430,25	1,2862	620,24	434,75	1,2997	454,94
5.2.20	334,5	433	1,2945	1225,22	423	1,2646	1219,48
5.2.30	334,5	425	1,2706	1817,1	419,5	1,2541	1816,57
5.2.40	334,5	425	1,2706	2416,16	417,75	1,2489	2419,16
5.2.50	334,5	426,75	1,2758	3019,4	424	1,2676	3019,04
5.2.60	334,5	421,25	1,2593	3537,36	424	1,2676	3617,06
5.3.3[1]	1337,5	1686	1,2606	196,48	1686	1,2606	196,48
5.3.10	1337,5	1626,25	1,2159	618,5	1630,5	1,2191	616,38
5.3.20	1337,5	1625,5	1,2153	1217,39	1597,75	1,1946	1220,54
5.3.30	1337,5	1617,25	1,2092	1817,65	1599,25	1,1957	1818,14
5.3.40	1337,5	1608,25	1,2024	2420,32	1592,25	1,1905	2419,72
5.3.50	1337,5	1600,25	1,1964	3018,13	1582,5	1,1832	3025,22
5.3.60	1337,5	1598,75	1,1953	3617,11	1575,25	1,1778	3616,75

[1] Instance does not use the column generation method, only the initial set of shifts is used in to find integer solutions

Table 26: Results time sensitivity ten filters

Number	Demanded (h)	CG exact pricing			CG heuristic		
		Result (h)	VDC	Time (s)	Result (h)	VDC	Time (s)
10.1.3[1]	994,5	1383	1,3906	213,89	1383	1,3906	213,89
10.1.10	994,5	1324,25	1,3316	742,51	1263,25	1,2702	636,55
10.1.20	994,5	1317	1,3243	1265,43	1243,25	1,2501	1241,67
10.1.30	994,5	1303,75	1,311	1846,4	1238,5	1,2453	1837,75
10.1.40	994,5	1287,5	1,2946	2479,02	1235	1,2418	2438,09
10.1.50	994,5	1278,75	1,2858	3111,11	1231,25	1,2381	3080,63
10.1.60	994,5	1282,25	1,2893	3670,06	1218,75	1,2255	3638,84
10.2.3[1]	1127,5	1699,5	1,5073	213,12	1699,5	1,5073	213,12
10.2.10	1127,5	1589,25	1,4095	689,13	1420,75	1,2601	638,62
10.2.20	1127,5	1549,75	1,3745	1313,08	1394,5	1,2368	1237,33
10.2.30	1127,5	1524,5	1,3521	1841,05	1398,25	1,2401	1836,91
10.2.40	1127,5	1489,75	1,3213	2495,12	1389,5	1,2324	2437,21
10.2.50	1127,5	1473,75	1,3071	3114,88	1372,25	1,2171	2290,16
10.2.60	1127,5	1469,25	1,3031	3676,1	1364,25	1,21	3641,31
10.3.3[1]	1312	1944,75	1,4823	214,84	1944,75	1,4823	214,84
10.3.10	1312	1827,25	1,3927	657,59	1660,75	1,2658	637,37
10.3.20	1312	1751	1,3346	1301,25	1631,25	1,2433	1238,55
10.3.30	1312	1743,5	1,3289	1851,29	1627,5	1,2405	1839,74
10.3.40	1312	1738	1,3247	2557,71	1620,5	1,2351	2437,52
10.3.50	1312	1713,25	1,3058	3043,92	1611,5	1,2283	3036,24
10.3.60	1312	1693,75	1,291	3655,77	1596	1,2165	3645,26

[1] Instance does not use the column generation method, only the initial set of shifts is used in to find integer solutions