

ERASMUS UNIVERSITY ROTTERDAM  
ERASMUS SCHOOL OF ECONOMICS  
Master Thesis Operational Research and Quantitative Logistics

---

Using Machine Learning predictions in Operations  
Research techniques to solve the Nurse Rostering  
Problem

Tim Tjhay (495230)

---



---

Supervisor:	P.C. Bouman, D.D. Tönissen (ORTEC), L. Berghman (ORTEC)
Second assessor:	W. van den Heuvel
Date final version:	9th November 2023

---

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

This thesis evaluates the viability of a solution method to the Nurse Rostering Problem that combines Machine Learning (ML) and Operations Research. Furthermore, several planning approaches are examined, to determine whether it could be interesting for hospitals to change their approach. For this method an ML model is used that predicts how good the assignment of a nurse to a duty is, which is created by ORTEC using data from a hospital that they are cooperating with. It is shown that a feasible schedule can not be created if the ML model is used by itself or in combination with simple heuristics. Therefore, these predictions are used in a Mixed Integer Program and a Simulated Annealing algorithm to create a schedule. Both of these methods are able to produce schedules that are comparable to the realized schedule in terms of quality, showing the potential of the approach. Especially the Mixed Integer Program seems to work well as it is able to find optimal solutions for realistic instances within several minutes. It also became clear that planning multiple departments simultaneously can drastically reduce the usage of flex nurses at the cost of a decrease in schedule quality.

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Literature Review</b>	<b>5</b>
2.1	Nurse Rostering Problem . . . . .	6
2.1.1	Overview . . . . .	6
2.1.2	Solution Methods . . . . .	7
2.2	ML in OR . . . . .	7
<b>3</b>	<b>Problem Description</b>	<b>8</b>
3.1	Background Information . . . . .	8
3.2	Aim of Research . . . . .	10
3.3	Nurse Rostering Constraints . . . . .	10
3.4	Case Study . . . . .	11
3.4.1	Data . . . . .	11
3.4.2	Case Specific Constraints . . . . .	13
3.4.3	ML model . . . . .	14
3.4.4	Extensions . . . . .	15
<b>4</b>	<b>Methodology</b>	<b>16</b>
4.1	Simple Heuristics . . . . .	16
4.2	Mixed Integer Program . . . . .	16
4.3	Metaheuristics . . . . .	20
4.3.1	Neighborhoods . . . . .	20
4.3.2	Simulated Annealing . . . . .	22
4.3.3	Variable Neighborhood Search . . . . .	24
4.4	Experimental Setup . . . . .	24
<b>5</b>	<b>Results</b>	<b>26</b>
5.1	Quality Created Schedule . . . . .	26
5.2	Alternative Planning Approaches . . . . .	28
5.2.1	Centralized Monthly Planning . . . . .	29
5.2.2	Decentralized Quarterly Planning . . . . .	31
5.2.3	Centralized Quarterly Planning . . . . .	32
5.3	Comparison Methods . . . . .	33
5.3.1	Time Until First Feasible Solution MIP . . . . .	33
5.3.2	Analysis Runs SA . . . . .	34
5.3.3	Comparison . . . . .	37
<b>6</b>	<b>Discussion</b>	<b>39</b>
<b>7</b>	<b>Conclusion</b>	<b>41</b>

<b>References</b>	<b>43</b>
<b>A Weights SA</b>	<b>46</b>
<b>B Evaluation Upper Bound</b>	<b>47</b>
<b>C Additional Results</b>	<b>48</b>

# 1 Introduction

Operations researchers have been studying employee scheduling for more than 50 years, with new developments being made every year. A specific case of this problem is the Nurse Rostering Problem (NRP), which is encountered by every hospital in the world. This problem aims to create a schedule for the nurses in the hospital that is optimal w.r.t. a certain objective while ensuring several constraints are satisfied. Often this schedule is still made manually, which is a long and complex process. Because of this, Machine Learning (ML) and Operations Research (OR) can be used to develop algorithms for the NRP, that are able to create a schedule automatically.

The NRP creates a schedule by assigning nurses to the different types of duties during the day. In many cases the day is divided into three duties based on the start and end times, with each day consisting of an early, late and night duty, however, this also depends on the hospital and the department. The assignment that is made has to satisfy several hard constraints that ensure that all patients can be taken care of and the nurses are not overworked. There are also several soft constraints that do not necessarily have to be satisfied, such as the preferences of the nurses. However, it is desirable to satisfy as many of them as possible, to create a schedule that is optimal for both the planners and the nurses.

Many hospitals in the Netherlands currently schedule their nurses using software called ORTEC Workforce Scheduling (OWS) (ORTEC, 2021) which is provided by ORTEC, the leading supplier of mathematical optimization software (ORTEC, 2023). However, ORTEC noticed that the planners at these hospitals often prefer to create a schedule manually and do not use the optimizer, that is included in the software and creates a schedule based on the given data. This could be an indication that there are soft constraints that have not explicitly been stated before creating the schedule or that are hard to express mathematically, which makes it difficult to include them in the optimization. Therefore, ORTEC decided to investigate an ML model that finds patterns in the historical data and detects these unknown soft constraints, in order to predict whether a nurse should be scheduled for a specific shift or not.

The idea behind these predictions is to encapsulate the unknown soft constraints and other preferences or patterns into scores that indicate how likely it is that both the planner and the nurse will be happy if a nurse is scheduled to work on a given day. ORTEC has currently built a prototype of the ML model based on the models developed by Cissen (2022) and Quak (2023). This model uses data on the realized schedules of a hospital in the Netherlands to predict these scores. Although these predictions might help in creating a schedule that is satisfactory for both the planners and the nurses, the predictions alone usually do not create a feasible schedule since none of the hard constraints are imposed by the model. In order to take these constraints into account while still using the insights of the ML model, several OR methods can be used.

Therefore, the aim of this thesis is to find out whether a solution approach that solves the NRP by using these scores in an OR method is a viable option, as this has not been researched in any of the found literature. It is also examined if this approach makes it possible to use different, more computationally expensive planning approaches, where either multiple months or multiple departments are planned simultaneously, and whether they are worth using instead of the current approach. Furthermore, because the scores are used as an input, no investigation is done into improving them; only the most effective way to use these scores is investigated.

To find an answer to these questions, the scores are used in a Mixed Integer Program (MIP) and a Simulated Annealing (SA) algorithm, after it was shown that a feasible schedule could not be created by using the ML model by itself or in combination with simple heuristics. The objective in these methods is to maximize the sum of these scores, as a schedule with a high total score should be better than one with a lower score. Since it is assumed that the soft constraints are encapsulated in the predicted scores, these do not have to be included in the objective. This seems to simplify the model compared to the ones presented in the existing literature, which typically use weighted sums of an often large set of soft constraints.

Because of this, the MIP performs quite well as most instances can be solved in less than five minutes. The MIP is also able to find schedules that have a significantly higher total score than the realized schedule. It also performs equally well on several KPIs that give an indication of the quality of a schedule. However, the MIP does struggle with very large instances, as it is not able to find any feasible solution for an instance consisting of over 150 nurses with a planning horizon of three months. To solve the problem for these large instances the SA algorithm can be used instead, which is able to find schedules that are at most 3% worse than the optimal solution. For smaller instances the metaheuristic performs even better, finding schedules that are within 1% of the optimal solution, although it does take more time than the MIP. Another thing to note is that the starting solution for this metaheuristic is also created using the predicted scores.

These results show the potential that this approach has and that this way of combining ML and OR is definitely a viable option that can be used to solve the NRP. It also became apparent that planning multiple months at once is likely not worth it as only a slight improvement can be made. On the other hand, planning multiple departments simultaneously makes it possible to drastically reduce the usage of flex nurses. However, since this does come at the cost of the quality of the schedule, it is up to the hospital to evaluate this trade-off and decide whether it is worth it.

The rest of this thesis is structured as follows: Section 2 contains an overview of the existing literature on the NRP and the use of ML in OR techniques. Afterwards, a detailed description of the problem is given in Section 3 as well as a description of the case that is studied. The methods used to solve the NRP are then presented in Section 4, after which the results are discussed in Section 5. Section 6 contains a discussion of the findings and, finally, the conclusions that are drawn from these findings can be found in Section 7.

## 2 Literature Review

In this section an overview is given of the existing literature that is relevant to either the NRP or the solution approach, that involves a combination of ML and methods from Operations Research (OR). In Section 2.1 a summary is given of how the NRP is formulated in the literature and how it is solved. Afterwards, the literature on the various ways that ML can be used in OR methods is discussed in Section 2.2.

## 2.1 Nurse Rostering Problem

Since employee scheduling is a problem that many companies encounter frequently, it has been studied extensively by operations researchers. Van den Bergh et al. (2013) and Ernst et al. (2004) both give an overview of the general problem, but while Van den Bergh et al. (2013) focus more on the most common constraints and solution methods, Ernst et al. (2004) focus more on the specific problems that are encountered in different employment sectors. The problem has also been shown to be NP-complete by Cooper and Kingston (1996), prompting researchers to develop metaheuristics to solve it.

In the health care sector, employees need to be scheduled during the night and during weekends as well, which makes the problem more complex. For this reason a lot of research is done on scheduling in this sector, with many studies focusing on the NRP in particular. In Section 2.1.1 an overview of the literature on the NRP is given and in Section 2.1.2 several methods that have been used to solve the problem are presented.

### 2.1.1 Overview

An overview of the existing literature on the NRP is given by Burke et al. (2004b), De Causmaecker and Vanden Berghe (2011) and Cheang et al. (2003), who all compare the formulations of the problem in different papers, which constraints are often included and the various solution methods that can be used. While the three papers have a lot of similarities, Burke et al. (2004b) focuses more on the practical aspect of the problem and how it fits in the more general context of employee scheduling in hospitals, De Causmaecker and Vanden Berghe (2011) aim to introduce a notation that would make it easier to apply methods for the NRP to more general problems and Cheang et al. (2003) focus more on the theoretical aspect of the problem such as the constraints that are taken into account and the objective that is formulated.

Some research has also been done to verify whether the theoretical results in the literature are representative of the real world. Drake (2014) surveyed several hospitals in Malaysia to examine the validity of the constraints that are often used. From the survey it became apparent that the constraints from the literature were indeed used, however, it could also be seen that several hard constraints were sometimes implemented as soft constraints. Additionally, Kellogg and Walczak (2007) showed that only 30% of the systems proposed in the literature are implemented by the hospitals, indicating that further research is needed to bridge the gap between the theoretical and practical solutions.

As the resulting schedule should satisfy both planners and nurses, Breugem et al. (2022) and Burke et al. (2001b) researched how a schedule should be evaluated. While Breugem et al. (2022) does not focus specifically on schedules for nurses, they do conclude that a balance needs to be found between the fairness and attractiveness in a schedule. The fairness refers to an equal distribution of the workload and attractiveness to how attractive the schedule is to each individual nurse, e.g. how many preferences are fulfilled and how many days off they get between shifts. To measure this attractiveness, Burke et al. (2001b) propose an algorithm that evaluates a schedule by examining the personal schedule for each individual nurse. This evaluation method was shown to be effective when using metaheuristics, in particular evolutionary algorithms, due to its speed and low memory usage.

### 2.1.2 Solution Methods

Since the NRP is a complex problem that can generally not be solved to optimality using standard solvers, it is often solved using metaheuristics that aim to find a good solution quickly, even if this solution is not optimal. An example of these metaheuristics is the Variable Neighborhood Search (VNS), which is used by Burke et al. (2004a) to solve the NRP. To improve the performance of metaheuristics, they can also be combined to create hybrid approaches, which is done by Burke et al. (2008), who combined the VNS with heuristic ordering. This method proved to be especially effective on smaller instances consisting of up to 20 nurses.

However, since large departments in hospitals often have more than 20 nurses, several other metaheuristics can be used to improve the solution, as is done by Goodman et al. (2009), who use a hybrid Greedy Randomized Adaptive Search Procedure (GRASP) and a knapsack approach to solve the problem. Another metaheuristic that can be used to avoid local optima is the Tabu Search (TS), which is used by Burke et al. (1998) and Bellanti et al. (2004). Burke et al. (1998) describe the TS algorithm that was developed for nurse rostering software for Belgian hospitals and Bellanti et al. (2004) present a TS as well as an Iterated Local Search.

Alternatively, Thompson (1996), Ceschia et al. (2023), Turhan and Bilgen (2020) and Hadwan (2022) all use Simulated Annealing (SA) to solve the NRP. The first two use only SA while the other two combine it with another method to create a hybrid approach. Thompson (1996) applies SA to a simplification of the problem where skill levels are ignored. On the other hand, Ceschia et al. (2023) does not simplify the problem and applies SA to several real-world instances from Italian health care institutions. Turhan and Bilgen (2020) describe an algorithm that combines the SA with the fix-and-optimize heuristic and Hadwan (2022) proposes a hybrid of SA with a Harmony Search.

Several evolutionary algorithms have also been used to solve the NRP as is done in Kelemci and Uyar (2007), Awadallah et al. (2015) and Wu et al. (2013), who use a Genetic Algorithm, an Artificial Bee Colony algorithm and Ant Colony Optimization respectively. Burke et al. (2001a) propose a hybrid of a TS and a Memetic Algorithm (MA) that is shown to outperform both approach when they are implemented individually. A MA is also used by Aickelin et al. (2007), who use it to obtain probabilities in their Estimation of Distribution Algorithm. Alternatively, mathematical programming based methods can be used such as Column Generation (CG) as is done in both He and Qu (2012) and Baeklund (2014).

## 2.2 ML in OR

In recent years OR and ML experts have researched methods that use ML to solve complex combinatorial optimization problems. These methods could use only ML to solve the problem or they could use ML as part of an algorithm or metaheuristic that is often used in OR. An example of the former is given by Vinyals et al. (2015), who use Recurrent Neural Networks to solve the Traveling Salesman Problem, while an example of the latter is given by Kruber et al. (2017) and Lodi et al. (2020) who both use ML to classify whether an algorithm could be applied to a given instance. Another way that ML can be used as a part of an algorithm is presented by Fitzpatrick et al. (2021) and Tayebi et al. (2022), who use ML to prune parts of the solution space that are unlikely to be optimal, before solving the problem using a standard solver. A



more extensive overview of the literature on ML in OR is given by Bengio et al. (2021).

Research has also been done into implementing ML into algorithms for scheduling problems. For example, Kumar et al. (2019) use ML to solve the NRP by learning constraints from historical schedules and solving the problem, with the learned constraints, using a standard solver. Besides the NRP, ML has also been used to solve the Crew Pairing Problem. Yaakoubi et al. (2020) and Tahir et al. (2021) both use ML to predict probabilities that show how likely it is that a pairing will be in an optimal solution and they use these probabilities in a CG algorithm. Yaakoubi et al. (2020) mainly use the predictions to obtain starting solutions for the CG and in several algorithms that are used to speed up the CG. On the other hand, Tahir et al. (2021) also use the predictions in the pricing problem of the CG, to ensure that no variables are added that are very unlikely to result in an optimal solution. Additionally, Tahir et al. (2021) use a score that is based on a ranking of the probabilities to determine which option to add, instead of using the actual probabilities, as they saw that it decreased the run time significantly. They argued that this decrease is caused by the fact that the score based on the ranking is discrete and the probabilities are continuous. In the continuous case, many alternative solutions with nearly identical values need to be evaluated as one of them might bring a slight improvement, even if this improvement is insignificant. However, in the discrete case, these alternatives will likely have the same score, which makes it unnecessary to evaluate them further. Therefore, the total number of evaluated solutions is reduced and the run time is decreased.

### 3 Problem Description

In this section a detailed description is given of the problem at hand and the case study that will be conducted to test the methods on real life data. Firstly, in Section 3.1 some background information is given on the problem. Then the goals of the research are discussed in Section 3.2, after which the constraints that need to be taken into account are presented in Section 3.3. Finally, Section 3.4 describes the case that is studied, giving some additional information specific to the hospital whose data is used, such as added constraints and assumptions.

#### 3.1 Background Information

ORTEC supplies their scheduling software, ORTEC Workforce Scheduling (OWS) (ORTEC, 2021), to many hospitals in the Netherlands. However, they noticed that the optimizer, that is included in OWS, is often not used and that a schedule is usually created manually by the planners at these hospital. A possible cause for this could be soft constraints that either have not explicitly been stated or are hard to express mathematically, which makes it difficult to include them in the optimization. Therefore, ORTEC decided to investigate whether machine learning (ML) could be used in an alternative solution approach that might perform better than the one used by the current optimizer. This investigation is done in cooperation with one of the hospitals that use OWS, who are providing historical data on the realized schedules of the nurses for the different departments.

In this hospital each day is split up into a day (D), evening (A) and night (N) duty and for each duty a capacity is determined at the start of the planning period. This capacity corresponds

to the number of nurses with a certain skill that should be working during that duty to ensure that patients get the care they need. In the rest of this report, a shift refers to one of the slots in a duty while a duty refers to all slots together, e.g. if a given duty has a duty capacity of three, there are three shifts of this duty that a nurse can be assigned to. The hospital in question currently plans each of their departments individually and independent of the other departments, which will be referred to as ‘decentralized planning’ in the rest of this thesis. Most of these departments use an approach called ‘self-scheduling’, which consists of three rounds. However, before this process starts, planners first compare the total contract hours for all nurses within a department with the total required hours in that department. If the required hours can not be filled using only the nurses within the department, flex nurses are assigned to the department and go through the same process as the other nurses in the department.

Once the flex nurses are added, the first round starts and the nurses define their own ideal schedule without regarding the capacity for any given duty. During this round each nurse can also formulate three mandatory preferences that have to be satisfied in the final schedule. Afterwards, the nurses can see the resulting schedule that is created by combining each of the personal schedules and the duties where the capacity is either not met or exceeded. Then, in the second round the nurses are allowed to swap shifts or change the duty that they have chosen a shift from, in order to fix the under- or overstaffing of these duties. For each swap that a nurse makes, they receive points indicating how flexible they were during this round. Finally, in the third round any remaining issues are fixed by a planner and the final schedule is published. While fixing these violations, the planners take the points from the second round into account, to ensure that nurses that swapped many of their preferred duties are not forced to make any other changes. Since the nurses can choose individual shifts when creating their own schedule in the first round, the final schedule is acyclical and each duty on every day is planned individually, which is why the same is done in the solution approach.

Currently, ORTEC has built a prototype of an ML model that finds patterns in the historical data on the realized schedules of a given department based on the work done by Cissen (2022) and Quak (2023). This model predicts how likely it is that a given nurse will be scheduled for a shift of a specific duty on a given day and assigns a score between one and zero to how ‘good’ the assignments, where a very good assignment has a score of one and a bad assignment has a score of zero. These scores should encapsulate the unknown soft constraints and other preferences or patterns and they should give an image of how likely it is that both the planner and the nurse will be happy with the assignment. An initial schedule can also be created based on these scores, which can be used as a starting solution for a metaheuristic that creates a feasible schedule. Alternatively, they could be used to prune the solution space by excluding nurse-to-duty assignments that are highly unlikely in an attempt to improve the performance of an algorithm. However, the predictions are only made for nurses within the department, which means the data on the shifts done by the flex nurses or nurses from different departments is filtered out and the external nurses need to be modeled in another way. Furthermore, the predicted scores are seen as an input for the developed methods and not as a part of the results. This is done because the focus in this thesis is on finding the most effective way to use the scores and not on how to improve them.

## 3.2 Aim of Research

The main goal of this thesis is to examine the viability of an approach that combines ML and OR to solve the NRP. To do this, the resulting schedules are compared to the realized ones, which should give an indication of the quality of these created schedules. This quality is also used to determine to what extent the OR side of the combination is actually needed and, if it is necessary, which OR method works best. In this comparison between methods, the runtime needed to create the schedule is not evaluated very thoroughly and it is only discussed briefly. This is due to the fact that the planners at the hospital stated that the runtime is not as important, since a schedule is only made once a month and they could run it overnight. For this reason, the runtime of methods is not investigated very thoroughly, as long as a schedule can be created within ten hours.

Besides the evaluation of the approach itself, research is also done into alternative planning approaches that might become possible due to the simplification of the model, caused by the encapsulation of multiple soft constraints into a single score using the ML model. For these alternative planning approaches, the benefits and drawbacks compared to the current situation are evaluated, as well as the effect that they have on the performance of the OR methods.

## 3.3 Nurse Rostering Constraints

Many constraints need to be taken into account when creating a schedule. An important part of creating a feasible schedule is to ensure that the coverage constraints are met, i.e. that the duty capacity is met for each skill level of each duty on every day. Besides these coverage constraints there are many labor laws that need to be adhered to, namely the Collective Labor Agreement (NL: *Collectieve Arbeidsovereenkomst*) and the Labor Time Law (NL: *Arbeidstijdenwet*). Since these laws consist of many constraints not all are taken into account during the modeling of the problem. Based on existing literature and the rules currently used in OWS the following subset is used in the model:

- Forward rotation, if a nurse works two days in a row, there needs to be at least 24 hours between the start times of the shifts, e.g. if they work an evening shift they can do an evening or night shift the next day but not a day shift.
- A nurse can work a maximum of 60 hours in a week
- A nurse can do at most 20 night shifts in 4 weeks
- A nurse needs to have at least 46 hours of rest after three or more consecutive night shifts
- A nurse can do at most five consecutive night shifts
- A nurse can do at most seven consecutive shifts if at least one of the shifts is a night shift
- A nurse needs to have at least 22 weekends off work in a year
- A nurse does not have to work exactly their contract hours in a week, but any deviations need to be balanced out at the end of the year

Because of these constraints it might not be possible to satisfy the coverage constraints for every duty. Therefore, the flex nurses, briefly mentioned in Section 3.1, can be used to meet the capacity. However, the use of these flex workers could cost the hospital more money which is why it would be ideal to use them only if there is no other alternative.

Many different schedules could be made that satisfy these constraints and for that reason, a criterion needs to be chosen that can be used to compare them. The criterion that is used in this thesis is the maximization of the total score of a schedule, which is the sum of the predicted scores of the assignments that are made in the evaluated schedule. By maximizing this, the unknown soft constraints, that should be encapsulated into the scores by the ML model, are taken into account in the eventual schedule. This criterion is the focal point of the solution approach that is investigated, as it is the link between the ML model created by ORTEC and the work done in this thesis.

### **3.4 Case Study**

To test the performance of the methods presented in this thesis in a real world setting, a case study is performed using data provided by the hospital that ORTEC is working with, which is discussed in Section 3.4.1. Afterwards, several additional constraints are presented in Section 3.4.2, that might not be applicable to other hospitals. The previous work done by ORTEC is then described in Section 3.4.3 and the alternative planning approaches that are evaluated are described in Section 3.4.4.

#### **3.4.1 Data**

In order to test the applicability of the approach in a real world setting, data of one of the hospitals that uses OWS is provided by ORTEC. This data was collected between 2016 and 2022 and contains information about the nurses in each department within the hospital, e.g. when they were under contract, their contract hours and their skill level. The predetermined capacity for each duty, previously described in Section 3.1, can also be found in the data. Besides this data, the realized schedules are available and show which nurses actually worked a shift of a specific duty, taking last minute changes such as sickness into account. However, the planned schedules that were published by the planners before any last minute changes are not available, since they are overwritten when a change is made and no copy is saved.

Since the planned schedules are not available, the realized ones are used in the ML model. However, this does cause some complications as there is no way to verify if the planners would still think the realized schedule is good, as it might have changed drastically since the original planned schedule was approved. Another complication that is caused by the use of realized instead of planned schedules is that the predetermined capacity for the duties is not always met in the realized version since it might not always be possible to find a replacement in case of last minute cancellations.

Ideally, the ML model would be trained on perfect information, however, since only the realized schedule is available, the assumption needs to be made that this schedule is perfect. This also means that the provided duty capacity is changed to match the number of nurses

that was assigned to the duty in the realized schedule with the assumption that this change in capacity was known beforehand.

During the development of the ML model the data of three different departments is used, namely the Intensive Care (IC) department, the Cardiology department and the Neurology department. The main reason for selecting these departments is their size as they consists of 68, 36 and 33 nurses, respectively, and they are, therefore, relatively large departments. Due to their size they provide the ML model with more data to train on which should help with getting accurate predictions. For this reason these three departments are used in this thesis as well, to test the applicability of the proposed approach to real world data. To show some of the differences between these departments, Figure 1 shows the number of shifts that is done by nurses within the department and the shifts that are done by nurses from other departments, e.g. the flex nurses.

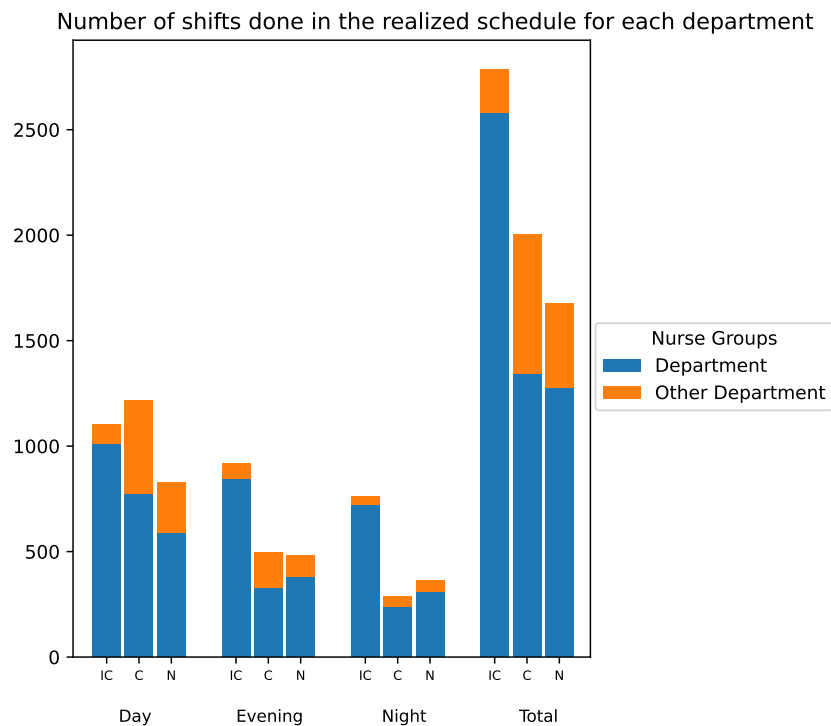


Figure 1: Number of shifts done by nurses within and outside of the department for each model, where C corresponds to Cardiology and N to Neurology

In the figure it can be seen that the IC department is the biggest department but that it is also the department that uses nurses from other department the least. On the other hand the figure shows that the Cardiology department uses nurses from other departments the most and that nearly a third of all shifts is done by one of these nurses. It can also be seen that there are significantly less evening and night shifts in the Cardiology and Neurology departments compared to the IC department.

Since there is no data on the availability of the nurses from the other departments, only the nurses within the department are planned and the shifts that the nurses from other departments do in the realized schedule are fixed. It is, therefore, assumed that these shifts are assigned to the nurses beforehand and that this is known when the schedule is created. The duty capacity

can then be lowered to take these assigned shifts into account as they no longer need to be done by a nurse within the department. This is done to mimic the circumstances that the realized schedule were executed in as closely as possible, in an attempt to make the comparison between the schedules as fair as possible.

As the hospital creates schedules on a monthly basis, the methods presented in this thesis are evaluated on the months of April, May and June of 2022. This period is chosen because this was the latest data that was available when the development of the ML model began. It will also be examined in a vacuum, which means that the schedules from before or after this period are not regarded. Additionally, the three months are split up based on full weeks, to make it easier to match contract hours. The three months consists of 91 days between April 1<sup>st</sup> and June 30<sup>th</sup>, which is equivalent to 13 weeks, with the first one spanning from April 1<sup>st</sup> to April 7<sup>th</sup>, the second starting on the 8<sup>th</sup> and ending on the 15<sup>th</sup> and so on. These weeks are then assigned to a planning month based on the date of the first day in the week, which results in the planning month of April consisting of 5 weeks, while May and June both consist of 4 weeks.

### 3.4.2 Case Specific Constraints

From discussions with the cooperating hospital it became apparent that several additional constraints are taken into account when creating the schedule on top of the labor laws. While these constraints might not apply to other hospitals, they are included into the methods evaluated in this thesis. The first of these added constraints is that a nurse needs at least 72 hours of rest after three or more nights shifts, opposed to the 46 hours that are required in the labor laws. Due to the combination of this constraint and the constraint on no more than 5 consecutive night shifts, it becomes impossible for a nurse to do more than 20 nights shifts in 4 weeks. This means that this constraint can be excluded from any models. Additionally, the hospital in question handles contract hours on a monthly basis, which means that a nurse should not work more than 17 hours, or 2 shifts, more than their contract hours in a month.

Another constraint that is taken into account by the planners, is that partial weekends should be avoided as much as possible. These are the weekends where a nurse works either on Saturday or Sunday but not on both days. A nurse should also not work more than three weekends in a row, and even working three consecutive weekends should be avoided if possible. Finally, the planners take the distribution of the evening and night shifts over the nurses into account. In this thesis this distribution is calculated using a formula that ORTEC is currently using in another hospital, namely:

$$\text{distribution } d = \sum_{n \in N} \left( v_n^d \right)^2, d \in \{A, D, \text{weekend}\} \quad (1)$$

Here,  $v_n^d$  is defined as the number of evening, night or weekend shifts that nurse  $n$  has worked in a given period.

### 3.4.3 ML model

Since the hospital is currently planning each department individually for one month at a time, the base case focuses on doing the same. This is also the case that was considered by ORTEC when creating the ML model. During this process the assumption was made that every shift of a given duty is the same length, where shifts of D and A duties are 8.5 hours long and shifts of N duties are 8 hours long and this assumption is, therefore, also made in this thesis. Because of this assumption the constraint on the weekly working hours in a week of a nurse can never be violated, which is why it is excluded from any models that are presented. Additionally, it is assumed that the duty capacity for skill level  $s$  does not need to be satisfied using only nurses with skill level  $s$ , but that it can also be filled using nurses that are more skilled.

The scores on nurse-to-duty assignments, that are predicted by the ML model, are also assumed to be correct. This means that they are able to encapsulate the preferences of both the planners and the nurses perfectly and that a schedule that consists of only assignments with very high scores is considered good by both planners and nurses.

In order to evaluate the scores obtained from the ML model, a schedule is created that can be used as a benchmark. In this schedule, each nurse is assigned the duty with the highest score for every day, where not working is also seen as a duty. This gives an upper bound on the total score that a schedule can have, which is why the resulting schedule is denoted as UB. However, it is likely that this schedule is not feasible since it does not take any of the hard constraints into account. In Table 1 the number of constraint violations for this schedule is compared to the violations in the realized schedule for the departments that are examined. In this table the labor laws are shown first, with the additional hospital specific constraints below them.

KPIs	IC		Cardiology		Neurology	
	RS	UB	RS	UB	RS	UB
Consecutive shifts	0	3	0	0	0	0
Consecutive shifts incl. night	2	15	1	1	0	0
Consecutive nights	10	13	0	0	0	1
Rest after night series	0	13	0	2	0	3
Forward rotation	121	248	17	44	8	52
Partial weekends	30	99	26	121	16	134
3+ consecutive weekends	20	46	12	28	11	28
Distribution evening shifts	5,579	6,201	2,080	1,108	2,507	2,342
Distribution night shifts	5,870	6,451	1,388	1,438	1,896	2,135

Table 1: Number of labor laws violations in schedules during April, May and June of 2022

It can be seen that the nearly all constraint are violated several times in the realized schedule, which can be caused by last minute changes. The figure shows that especially the forward rotation constraint is violated often in both the UB schedule and the realized schedule. This is especially true for the IC department as this law is broken more than 100 times in the realized schedule and nearly 250 times in the UB schedule. For all additional constraints it can also be seen that the UB schedule contains more violations than the realized one for at least one of the departments.

### **3.4.4 Extensions**

Besides the base case where a monthly schedule is made decentralized, several alternative planning approaches are examined in this thesis. For the first of these alternatives, the schedule is made on a quarterly basis instead of a monthly basis, which is described in Section 3.4.4.1. Afterwards, in Section 3.4.4.2 a planning approach is discussed where multiple departments are planned simultaneously.

#### **3.4.4.1 Decentralized Quarterly Planning**

Several labor laws are based on the yearly schedule of nurses and including these constraints in a shorter planning period often requires a more restrictive constraint, e.g. taking the minimum of 22 weekends off in a year into account in a monthly planning is done by imposing a minimum of 2 weekends off each month which might not be optimal. Because of this, planning longer periods at a time allows for more freedom in how these yearly constraints are satisfied with a yearly planning being the ideal scenario. On the other hand, nurses might not like it if schedules are made for such a long period, since it means they would have to register their availability and holidays more than a year in advance. For this reason, a planning horizon of three months is chosen as this still relaxes the restrictiveness of the yearly constraints, but is also not too long for the nurses.

#### **3.4.4.2 Centralized Planning**

Besides the decentralized planning approach, this thesis also considers a hypothetical case where departments are planned simultaneously, which is called centralized planning. In this scenario, there is a flex pool consisting of a number of flex nurses that need to be planned, instead of assuming that the planners plan them manually before the rest of the schedule is created. This means that the shifts that are filled by flex nurses are actually assigned to a specific flex nurse by the model, which makes it possible to ensure no labor laws are violated for the flex nurses. It also becomes possible to include the exchange or borrowing of nurses between departments when planning. This can help when there is one department that does not have enough nurses to meet their duty capacity and another department that has too many nurses, which means it is not possible to meet the contract hours for all of them. Another advantage of centralized planning is that meetings involving nurses from multiple departments could be planned easier.

When considering centralized planning, several additional assumptions need to be made. Since it is not known in which departments each flex nurse can work and what their skill level would be in a given department, the assumption is made that a flex nurse can work any duty, regardless of the required skill level and the department. Additionally, no score is gained from assigning flex nurses to duties, while assigning a flex nurse not to work on a day results in a score of 1. This is done to model the fact that these flex nurses should only be used if there is no other option. It is also assumed that any nurse can be borrowed by another department and that the skill level of a nurse is lowered by one if they are borrowed. Finally, it is assumed that nurses do not like to work in other departments, which is modeled by halving the score gained from an assignment of a nurse to a duty if it is a duty in a different department.



In this experiment, the case is studied where the three departments, presented in Section 3.4.1, are planned simultaneously with a flex pool consisting of 20 flex nurses who have a contract for 30 hours in a week. This number is chosen based on the total number of shifts that is done by flex nurses in the realized schedule for the three departments. Since this is data from the same hospital, the additional constraints mentioned in Section 3.4.2 are still included. Furthermore, this centralized approach is used to create both monthly and quarterly schedules, as is done for the decentralized one. The quarterly planning approach could also give an indication of how the methods react to very large instances for the monthly approach, where more than three departments need to be planned simultaneously.

## 4 Methodology

In this section the methods are described that are used to go from the scores, predicted by the ML model, to a feasible schedule. First, several simple heuristics are presented in Section 4.1, after which a Mixed Integer Program (MIP) is formulated in Section 4.2. Two metaheuristics are then described in Section 4.3 and, finally, Section 4.4 contains a description of the experimental setup, i.e. on what aspects the schedules are compared and how these comparisons are interpreted.

### 4.1 Simple Heuristics

Because the UB schedule is infeasible, as was shown in Section 3.4.3, two simple heuristics are used to remove these violations. The first one, called Heuristic 1 (H1) uses the UB schedule as a starting point and iteratively removes the shifts that are causing any infeasibilities. Afterwards an attempt is made to fill the duty capacity by assigning nurses who can work the duty while maintaining the feasibility.

The second heuristic, called Heuristic 2 (H2), works in a similar way, with the only difference being the starting solution. For this second approach, an initial schedule is created by filling the duty capacity,  $C_{td}^s$ , using the nurses with the highest probability for the given duty. To find these nurses a list is constructed of all nurses that have a non-zero probability of doing the duty and that are not assigned to another duty on the same day. After which the  $C_{td}^s$  nurses on the list with the highest probability are assigned to the duty. However, if the number of nurses on the list is lower than the duty capacity, all nurses on the list are assigned to the duty and the remainder is seen as understaffing. For every day, this is done for the night, evening and day duty consecutively. This order is chosen since it might be easier to find a flex nurse that is willing to do a day duty than one that is willing to do a night duty. The same process that was used to make the schedule feasible in H1 is then used to make this schedule feasible.

### 4.2 Mixed Integer Program

Based on the constraints from Section 3.3 and Section 3.4.2 a MIP is formulated in (2a)-(2r).

**Sets:**

- $N$  is the set of nurses that work at the department
- $T$  is the set of the days in the planning horizon
- $W$  is the set of weeks in the planning horizon

- $T_w$  is the set of days in week  $w$
- $M$  is the set of months in the planning horizon
- $M^0$  is the set of months in the planning horizon and the month prior to the start of the planning horizon,  $m^0$
- $T_m$  is the set of days within month  $m$
- $T^0$  is the set of the days in the planning horizon and  $m^0$
- $W^0$  is the set of the weeks in the planning horizon and  $m^0$
- $D^0$  is the set of all duties, where not working corresponds to 0, a day duty to 1, an evening duty to 2 and a night duty to 3
- $D \subset D^0$  is the set of working duties where the not working duty is excluded
- $S$  is the set of skill levels that a nurse can belong to where 0 is the highest and can do duties of all lower skill levels, 1 is the second highest and can do all duties except for the ones of skill level 0 and so on
- $N^s$  is the set of nurses that are allowed to do duties in skill level  $s$ , i.e. the nurses with skill level  $s$  or better

#### Parameters:

- $p_{ntd}$  is the score that is gained if nurse  $n$  is assigned to work duty  $d$  on day  $t$
- $l_k$  is the length of duty  $k$
- $C_{td}^s$  is the required capacity for nurses of skill level  $s$  during duty  $d$  of day  $t$
- $h_{nm}$  is the number of hours that nurse  $n$  should work in the weeks that start in month  $m$  according to their contract
- $K_n$  is the advised minimum number of weekends off that nurse  $n$  should have in the planning period that is derived from the yearly minimum of 22, e.g. for a planning horizon of the three months  $K_n = \frac{22}{4} = 5.5$ . The number of weekends off nurse  $n$  has had in the previous planning periods in the year can also be taken into account.
- $P_{\max}$  is the maximum number of times a nurse works a partial weekend in the schedule
- $V_{\text{consec}}$  is the maximum number of consecutive weekends that a nurse is allowed to work
- $V_{\max}$  is the maximum number of times a nurse is allowed to work  $V_{\text{consec}}$  weekends in a row
- $U_d$  is an upper bound on the distribution for duty  $d$
- $x_{ntd}^*$  is used to set the schedule for  $m^0$  and is equal to 1 if nurse  $n \in N \cup F$  did duty  $d$  on day  $t$  and 0 otherwise

#### Decision variables:

- $x_{ntd}$  is a binary variable that is equal to 1 if nurse  $n \in N \cup F$  is planned to work duty  $d$  on day  $t$
- $R_{nw}$  is a binary variable that is equal to 1 if nurse  $n$  works during the weekend in week  $w$
- $P_{nw}$  is a binary variable that is equal to 1 if nurse  $n$  works a partial weekend in week  $w$
- $V_{nw}$  is a binary variable that is equal to 1 if nurse  $n$  works  $V_{\text{consec}}$  weekends in a row starting in week  $w$

$$\text{maximize } \sum_{n \in N} \sum_{t \in T} \sum_{d \in D^0} p_{ntd} x_{ntd} \quad (2a)$$

$$\text{subject to } \sum_{d \in D^0} x_{ntd} = 1 \quad n \in N, t \in T^0 \quad (2b)$$

$$\sum_{n \in N^s} x_{ntd} \geq C_{td}^s \quad t \in T^0, d \in D, s \in S \quad (2c)$$

$$\sum_{t'=t}^{t+9} \sum_{d \in D} x_{nt'd} \leq 9 \quad n \in N, t \in T^0 \quad (2d)$$

$$\sum_{t'=t}^{t+7} x_{nt'3} + 7 \sum_{t'=t}^{t+7} \sum_{d \in D} x_{nt'd} \leq 56 \quad n \in N, t \in T^0 \quad (2e)$$

$$\begin{aligned}
& \sum_{t'=t}^{t+5} x_{nt'3} \leq 5 & n \in N, t \in T^0 & \quad (2f) \\
& \sum_{d \in D} \sum_{t'=t}^{t+2} x_{ntd} + 3 \sum_{t'=t-3}^{t-1} x_{nt'3} + x_{nt3} \leq 9 & n \in N, t \in T^0 & \quad (2g) \\
& x_{ntd} + \sum_{d'=1}^{d-1} x_{n(t+1)d'} \leq 1 & n \in N, t \in T^0, d \in D & \quad (2h) \\
& \sum_{t \in T_m} \sum_{d \in D} l_d x_{ntd} \leq h_{nm} + 17 & n \in N, m \in M & \quad (2i) \\
& \sum_{d \in D} x_{n(\text{sat}_w)d} + x_{n(\text{sun}_w)d} \leq 2R_{nw} - P_{nw} & n \in N, w \in W^0 & \quad (2j) \\
& \sum_{w \in W} R_{nw} \leq |W| - K_n & n \in N & \quad (2k) \\
& \sum_{n \in N} \sum_{w \in W} P_{nw} \leq P_{\max} & & \quad (2l) \\
& \sum_{w'=w}^{w+V_{\text{consec}}} R_{nw} \leq V_{\text{consec}} & n \in N, w \in W^0 & \quad (2m) \\
& \sum_{w'=w}^{w+(V_{\text{consec}}-1)} R_{nw} - V_{nw} \leq V_{\text{consec}} - 1 & n \in N, w \in W^0 & \quad (2n) \\
& \sum_{n \in N} \sum_{w \in W} V_{nw} \leq V_{\max} & & \quad (2o) \\
& \sum_{n \in N} \sum_{m \in M} \left( \sum_{t \in T_m} x_{nt2} \right)^2 \leq U_d & d \in \{2, 3\} & \quad (2p) \\
& x_{ntd} = x_{ntd}^* & n \in N, t \in T_{m^0}, d \in D & \quad (2q) \\
& x_{ntd}, R_{nw}, P_{nw}, V_{nw} \in \mathbb{B} & n \in N, t \in T, d \in D, w \in W & \quad (2r)
\end{aligned}$$

Here, the objective in (2a) corresponds to the maximization of the score in the schedule and ensures that the resulting schedule stays as close to the original predictions as possible. Constraint (2b) ensures that every nurse is either working a duty or has the day off for every day in the planning horizon. Constraint (2c) ensures that the capacity is met for all skill levels of every duty. Constraint (2d) imposes the maximum number of consecutive shifts a nurse can do, which is set to nine, and (2e) imposes a maximum of seven for series that include a night shift. Then, Constraint (2f) corresponds to the maximum number of five consecutive night shifts. Constraint (2g) ensures that a nurse gets at least three rest days after more than three consecutive night shifts and (2h) imposes the forward rotation constraint. Constraint (2i) ensure that each nurse does not work more than 17 hours, i.e. 2 shifts, more than their contract hours in a month.

Furthermore, the days  $\text{sat}_w$  and  $\text{sun}_w$  in constraint (2j) correspond to the Saturday and Sunday of week  $w$ , respectively. This constraint sets the variable that indicates whether a nurse works during the weekend of a given week or not and the variable showing whether a nurse works a partial weekend during a week. Constraint (2k) ensures that each nurse  $n$  has at least  $K_n$  weekends off and (2l) imposes the maximum on the number of partial weekends in the schedule. Constraint (2m) then ensures that a nurse does not work more weekends in a row than the maximum number of consecutive weekends. Moreover, the variable that indicates whether a nurse works the maximum allowed consecutive weekends is set using (2n), while (2o) sets the

upper bound on the number of times this occurs. The upper bounds on the distribution of evening and nights shifts is imposed by (2p) and the schedule for the month before the planning horizon,  $m^0$ , is fixed using (2q) to ensure no violations are made during the transition from the end of the last month to the start of the planning horizon. This is only done for May and June of 2022, since the period between April and June in 2022 is examined in a vacuum and it is therefore assumed there is no schedule from March. Finally, constraints (2r) determine the domain of the decision variables.

In this formulation the nurse-day combinations where a nurse has a score of 1 if they do not work are pruned and, therefore, not included in the model. This is done under the assumption that a nurse could never be assigned to a work duty one of these days in an optimal solution, which would make it unnecessary to try to optimize the assignment on these days.

In an ideal scenario, the parameters  $P_{\max}$  and  $V_{\max}$  would be set to zero since partial weekends and many consecutive weekends should not occur in the schedule. However, since only the realized schedule is available and it does contain these violations, the parameters are set to the number of times it occurs in the realized schedule. This is also done for  $U_2$  and  $U_3$ , in an attempt to make the comparison fairer and to recreate the circumstances that the realized schedule was executed in as closely as possible.

This same formulation can be used for the quarterly planning approach, however, for the centralized planning case the following slight changes need to be made to the MIP. The sets  $N$  and  $S$  are defined separately for each department  $a \in A$  where  $A$  is the set of departments including a dummy department for the flex nurses. Since all nurses can be borrowed, they can all do duties in any of the departments. Therefore, the duty set is defined as the union of the sets for all other departments, i.e.  $D = \cup_{a \in A \setminus \{\text{flex}\}} D_a$ . By substituting the sets in (2a)-(2r) with their department specific counterpart, e.g. substituting  $n \in N$  with  $n \in N_a, a \in A$  and  $\sum_{n \in N}$  with  $\sum_{a \in A} \sum_{n \in N_a}$ , the MIP can be used for this experiment.

As a result, the interpretation of the objective can be split up into two terms, namely, the total score that the schedule gains from the assignments of the nurses within the planned departments and the ‘score’ that is gained from the flex nurses. Since the flex nurses all have a score of 1 if they do not work on a day and 0 otherwise, this second term only depends on the number of shifts that are done by flex nurses. Since this number is easier to interpret than the corresponding score, it is used in the results instead of the score. Additionally, because of the added flex nurses and the ability to exchange nurses between departments, constraint (2c) is replaced by the following constraint:

$$\sum_{n \in N_a^s \cup N_{\text{flex}} \cup N_{\text{borrow}}^s} x_{ntd} \geq C_{td}^s, \quad a \in A \setminus \{\text{flex}\}, t \in T, d \in D_a, s \in S_a \quad (3)$$

Where,  $N_{\text{flex}}$  is the set of flex nurses and  $N_{\text{borrow}}^s$  is the set of nurses that can be borrowed for skill  $s$ . Additionally,  $P_{\max}$ ,  $V_{\max}$ ,  $U_2$  and  $U_3$  are set to the sum of these parameters for the different departments. For the partial weekends it, for example, becomes  $P_{\max} = \sum_{a \in A} P_{\max}^a$ , where  $P_{\max}^a$  is the value of the parameter that is used when only planning department  $a$ .

### 4.3 Metaheuristics

The scores are also used in two metaheuristics. In these metaheuristics a local search algorithm is used to improve the initial solution, which in this case is UB. For this metaheuristic a schedule is represented as a matrix, where the columns correspond to the days in the planning horizon and the rows to the nurses. The values in this matrix then correspond to the duty that the nurse does on the given day. An example of this representation of a schedule is shown in Figure 2.

	t <sub>1</sub>	t <sub>2</sub>	t <sub>3</sub>	t <sub>4</sub>	t <sub>5</sub>
n <sub>1</sub>		D	A		A
n <sub>2</sub>	D	A		A	N
n <sub>3</sub>	A		N	N	
n <sub>4</sub>		D	D	D	


Figure 2: Example of a representation of a schedule in the heuristic where rows correspond to nurses, columns to days and each cell corresponds to the duty that the nurse does on the day

The neighborhoods that are used in the local search are discussed in Section 4.3.1, while Section 4.3.2 gives a description of the Simulated Annealing (SA) algorithm that is used. Finally, in Section 4.3.3 a Variable Neighborhood Search is presented.

#### 4.3.1 Neighborhoods

Six neighborhoods are used in this local search that are inspired by neighborhoods that ORTEC currently uses in their optimizer and the ones that are often used for the NRP, such as in the work done by Ceschia et al. (2023). The first two neighborhoods consist of solutions that can be obtained by swapping a single shift between either two or three nurses, as long as the shifts are of different duties, and they are therefore called *DutySwap2* and *DutySwap3*. These swaps can be between working shifts or between at least one working shift and at least one day off. In Figure 3 two examples are shown of possible swaps in *DutySwap2*, one where the swapped shifts are on the same day and one where they are on different days.

	t <sub>1</sub>	t <sub>2</sub>	t <sub>3</sub>	t <sub>4</sub>	t <sub>5</sub>
n <sub>1</sub>		D	A		A
n <sub>2</sub>	D	A		A	N
n <sub>3</sub>	A		N	N	
n <sub>4</sub>		D	D	D	




	t <sub>1</sub>	t <sub>2</sub>	t <sub>3</sub>	t <sub>4</sub>	t <sub>5</sub>
n <sub>1</sub>		A	A		A
n <sub>2</sub>	D	D		A	N
n <sub>3</sub>	A		N	N	
n <sub>4</sub>		D	D	D	

(a) Example of same day swap

Figure 3: Examples of neighbors in *DutySwap2*

	t <sub>1</sub>	t <sub>2</sub>	t <sub>3</sub>	t <sub>4</sub>	t <sub>5</sub>
n <sub>1</sub>		D	A		A
n <sub>2</sub>	D	A		A	N
n <sub>3</sub>	A		N	N	
n <sub>4</sub>		D	D	D	



	t <sub>1</sub>	t <sub>2</sub>	t <sub>3</sub>	t <sub>4</sub>	t <sub>5</sub>
n <sub>1</sub>		D	A	N	
n <sub>2</sub>	D	A		A	N
n <sub>3</sub>	A		N		A
n <sub>4</sub>		D	D	D	


(b) Example of different day swap

Figure 3: Examples of neighbors in *DutySwap2* (cont.)

For the swaps where the shifts are on different days in *DutySwap2*, the shifts can be at most five days apart in an attempt to keep the running time low. A swap can also only occur if the nurses are not working on the day of their new shift, e.g. for the example in Figure 3b the swap would not be possible if it would involve  $t_3$  instead of  $t_4$  since  $n_1$  is already doing an evening shift on that day. The logic behind the swaps in *DutySwap3* works in a similar way, however, only the same day swaps shown in Figure 3a are considered. Additionally, in *DutySwap3* only the swaps where all three nurses are doing a different shift are considered, as a swap where at least two of the nurses are doing the same shift would be the same as a swap from *DutySwap2*. For example, a swap between  $n_1$ ,  $n_2$  and  $n_4$  on  $t_2$  in the schedule in the figure would result in the same schedule as a swap from *DutySwap2* between  $n_2$  and either  $n_1$  or  $n_4$ . While all possible pairs of nurses and all days are evaluated for *DutySwap2*, only a random sample of 500 combinations of nurses is evaluated for *DutySwap3*. This means that the time complexity for *DutySwap2* is  $\mathcal{O}(|N|^2|T|)$ , while it is  $\mathcal{O}(|T|)$  *DutySwap3*.

For the third and fourth neighborhood, named *SeriesSwap2* and *SeriesSwap3*, the neighboring solutions are obtained by swapping a series of shifts between two or three nurses. Here, a series of shifts for nurse  $n$  is defined as a subset of consecutive days for which it holds that  $n$  works on every day within this subset and does not work on the days right before and right after this subset. An example of this is given in Figure 4.

	t <sub>1</sub>	t <sub>2</sub>	t <sub>3</sub>	t <sub>4</sub>	t <sub>5</sub>
n <sub>1</sub>		D	A		A
n <sub>2</sub>	D	A		A	N
n <sub>3</sub>	A		N	N	
n <sub>4</sub>		D	D	D	



	t <sub>1</sub>	t <sub>2</sub>	t <sub>3</sub>	t <sub>4</sub>	t <sub>5</sub>
n <sub>1</sub>		D	A		A
n <sub>2</sub>	D	D	D	D	N
n <sub>3</sub>	A		N	N	
n <sub>4</sub>		A		A	

Figure 4: Example of neighbor in *SeriesSwap2*

In this figure,  $t_2$ ,  $t_3$  and  $t_4$  form a series for  $n_4$  which is why the shifts on these days can be swapped with  $n_2$ . As can be seen in the example, it is not necessary that the swapped period is a series for all nurses involved in the swap, e.g.  $t_2$ ,  $t_3$  and  $t_4$  do not form a series for  $n_2$  because they do not work on  $t_3$  and do work on  $t_1$  and  $t_5$ , but the swap is still allowed. This is done because the number of possible swaps would be significantly decreased if those swaps were not

allowed. Therefore, a swap is considered as long as the period is a series for one of the nurses and the series is longer than one day. For these two neighborhoods the same evaluation tactic is used as for the *DutySwap* neighborhoods and, since  $\frac{|T|}{2}$  is an upper bound on the number of series a nurse can have, the time complexity is the same as well.

Solutions in the fifth and sixth neighborhood can be obtained by adding or removing a single shift, respectively, which is why they are called *Add* and *Remove*. In *Add* a given nurse is assigned to one of the shifts of a duty on day on which they are not working. An example of this in terms of the solution representation in Figure 2 could be adding a D on  $t_1$  for  $n_1$ . For the neighbors in *Remove* a given nurse is given a day off one of the days on which they were previously working, e.g. the D on  $t_2$  is removed for  $n_1$  in the schedule shown in Figure 2. In the worst case of both neighborhoods all days need to be evaluated for all nurses, which means the time complexity is  $\mathcal{O}(|N||T|)$  for both *Add* and *Remove*.

In an attempt to escape local optima, the moves that are made in the six neighborhoods do not necessarily need to result in a feasible schedule. This means that during the algorithm it will be allowed to accept infeasible solutions, however, a violation of one of the hard constraints will introduce a weighted penalty term into the objective. This penalty term should ensure that any infeasibilities are eventually repaired and the resulting schedule is in fact feasible. This is also done by Ceschia et al. (2023) who modeled most of the constraints, that are considered hard in this thesis, as soft constraints for their Simulated Annealing heuristic.

### 4.3.2 Simulated Annealing

One of the local search algorithms that is used to improve the schedule is the Simulated Annealing (SA) algorithm, which was first introduced in Kirkpatrick et al. (1983) and applied to the NRP by Ceschia et al. (2023). Similar to other local search based heuristics, SA starts from an initial solution and looks for improvement by iteratively making small changes. However, it does incorporate diversification into this local search. To regulate how much diversification will be applied during the algorithm, SA introduces a temperature and worse solutions may be accepted during the algorithm based on this temperature, in an attempt to broaden the search. For this reason both the current solution and the best found solution in the previous iterations is stored during the algorithm. The temperature that is used in SA is initialized at the maximum temperature,  $T_{\max}$ , and is decreased by a cooling factor,  $\alpha$ , until it reaches the end temperature,  $T_{\text{end}}$ . This decrease takes place after a certain number of iterations, however, it could also be decreased if a new solution is accepted a certain number of times, which is also done by Ceschia et al. (2023). In this thesis, 150 and 75 are used for these parameters. When  $T_{\text{end}}$  is reached and no new best solution has been found for 75 iterations, it is assumed that the algorithm has found a local optimum and the temperature is therefore reset to a higher number in an attempt to escape this local optimum, which is also done by Osman (1995) who apply SA to the generalized assignment problem. In this thesis there is a maximum number of resets that can occur after which the algorithm can terminate. The number that the temperature is reset to becomes smaller with each reset, with the  $n^{\text{th}}$  reset setting the temperature to  $\frac{T_{\max}}{2^n}$ .

During each iteration of the algorithm one of the neighborhoods is randomly selected and from this random neighborhood a solution is chosen as a candidate to be the next solution.

The neighbor that is chosen is either a random solution in the neighborhood with probability  $0.1 + 0.8 \frac{T - T_{\text{end}}}{T_{\text{max}} - T_{\text{end}}}$  or the first improvement that is found when evaluating the neighborhood in a random order. By choosing this probability, random neighbors are chosen more often in the first stages of the algorithm and more improvements are chosen during the later iterations, but there is still at least a 10% chance of getting either option. This is done to allow the algorithm to diversify more in the earlier iterations and to force the algorithm towards the local or global optimum in the later iterations. Additionally, if the temperature is equal to the end temperature and the first improvement in the given neighborhood is chosen, there is a 5% probability of looking for the best improvement in the neighborhood instead of the first. This is done in an attempt get to the local optimum faster, however, this is not done very often, since looking for the best improvement can be quite expensive in terms of computing time.

If the candidate has a better objective than the current solution, it is always accepted and it becomes the new current solution. However, if the objective of the candidate is lower, then the candidate is accepted with probability  $\exp(\frac{\Delta f}{T})$ , where  $\Delta f$  is the difference in objective between the current solution and the candidate. Since the objective of the candidate is lower, this difference will always be negative, which means that the probability will always be between 0 and 1. These iterations are repeated until either a maximum number of iterations is reached or  $T_{\text{end}}$  is reached, no more resets are allowed and no new best solution is found for 100 iterations. After this, the algorithm is terminated and the best found solution is returned.

Additionally, the weights, that determine how heavily violations of hard constraints are penalized, are adjusted during the algorithm to prevent the weights from either being too high or too low, since a weight that is too high will restrict the traversal of the solution space and a very low weight will result in too many violations. This is also done by Vidal et al. (2012), who apply this weight adjustment to regulate the proportion of feasible solutions in the population for their Hybrid Genetic Algorithm that is applied to several Vehicle Routing Problems. The adjustment scheme that is used in this article is used in this algorithm, i.e. if there are either too many or too few violations the weight is multiplied by 1.2 or 0.85, respectively.

This adjustment does not occur before a nearly feasible solution is found. This is done since the starting solution, i.e. UB, contains a large number of violations and adjusting the weights while removing these violations would cause the weights to increase drastically, even though this would not be necessary. This means that the initial weights do still need to be chosen correctly, as they should be high enough to ensure that the SA will move to a nearly feasible solution. The weights will also not be adjusted when the temperature is above a given threshold, since a majority of the moves that are made when the temperature is high will be random moves. This means that violations can likely not be removed effectively and the weights will again be increased unnecessarily.

After a nearly feasible solution is found, the weights are updated every 500 iterations. To determine whether any adjustments need to be made during one of these updates, the average number of times each hard constraint is violated in the last 200 iterations is computed. Each of these averages is then compared to two predetermined thresholds, corresponding to a lower and upper bound on the average number of violations for the given constraint. If this average is lower than the first threshold the weight is multiplied by 0.85 and if it is higher than the



second threshold the weight is multiplied by 1.2. This average is only computed for the last 200 iterations to give the algorithm some time to adjust to the new weights.

### 4.3.3 Variable Neighborhood Search

Another metaheuristic that is used to improve on the initial schedule is the Variable Neighborhood Search (VNS). In this algorithm the neighborhoods are explored in order of their size, which, for the neighborhoods introduced in Section 4.3.1, is *Add-Remove-SeriesSwap3-DutySwap3-SeriesSwap2-DutySwap2*. Although *Add* and *Remove* have a worse theoretical complexity than *SeriesSwap3* and *DutySwap3*, they are smaller in practice which is why this order is chosen. This is caused by the fact that the number of samples that is taken for the latter neighborhoods is significantly larger than the number of nurses in the instance. For each of the neighborhoods the best neighbor is found and if this neighbor is an improvement on the current solution, then the bigger neighborhoods are not evaluated and this improvement is accepted. This metaheuristic is mainly used to test whether the diversification added by using SA is working properly and allowing the algorithm to escape or avoid local optima.

## 4.4 Experimental Setup

To evaluate whether these methods are a viable option for the NRP, several criteria are used to compare the resulting schedules to UB and the realized schedule. The first of which is the total score of the schedule, i.e. the sum of the scores gained by assigning shifts to nurses, which is the objective for all methods. Under the assumption that the predicted scores are correct, a schedule with a higher score should be better than a schedule with a lower score. Since UB is an upper bound on this score, this schedule can be seen as the perfect schedule according to the ML model. By looking at how close the total score of a schedule is to the score of UB, the quality of the schedule according to this model can then be determined.

Additionally, the F1-score w.r.t. the realized schedule is used as a criterion, since the assumption is made that the realized schedule is perfect. This F1-score is often used to evaluate the quality of classification models in ML, because it incorporates both the precision and recall of the model and, therefore, gives a more complete indication of how good the model is. In this thesis the F1-score is used to measure how close the schedules created by the methods are to this ‘perfect’ schedule. The values presented in this thesis are all obtained using the scikit-learn library (Pedregosa et al., 2011) for Python. To use this library the schedules are formatted as a solution to a multiclass classification problem, where the evaluated schedule can be seen as the predicted classifications and the realized schedule corresponds to the correct classifications. The following formulas are then used to compute the F1-score for each class of the multiclass classification problem, which in this case corresponds to each duty  $d \in D^0$ :

$$F1_d = \frac{2 \times precision_d \times recall_d}{precision_d + recall_d}, \text{ where } precision_d = \frac{TP_d}{TP_d + FP_d}, \quad recall_d = \frac{TP_d}{TP_d + FN_d} \quad (4)$$

In these formulas  $TP_d$  is the number of true positives, which is the number of nurse-day combinations, i.e. the cells of the matrix in Figure 2, where both the evaluated and the realized

schedule contain duty  $d$  in the cell. Furthermore, the number of false positives,  $FP_d$ , is the number of cells that contain duty  $d$  in the evaluated schedule but contain another duty in the realized schedule. Finally, the false negatives,  $FN_d$ , correspond to the number of cells where the realized schedule contains duty  $d$  and the evaluated schedule contains another duty.

After computing this F1-score for all duties, the scores can be averaged using several methods. In this thesis three averaging methods are presented, namely micro, macro and weighted averaging. In micro averaging the multiclass aspect is ignored and the total values over all duties are used for TP, FP and FN, which causes the F1-score to be equal to the accuracy of the model, i.e. the proportion of samples that are correctly classified. Alternatively, in macro averaging the unweighted mean is taken of the F1-scores of the duties, while a weighted mean is taken for the weighted F1-score. The scores of the duties are weighed based on the number of occurrences of the duty in the realized schedule in an attempt to take class imbalance into account.

Several KPIs are used as a third criterion in the comparison as they can help to get an idea of how good the schedules are. This is done since a schedule could still be good even if it is completely different from the realized schedule and even if it does not have a high total score. The performance of a schedule on these KPIs is compared to the performance of the realized schedule, which should give an indication of how the schedule compared to the realized one. For this comparison the following KPIs were selected by the cooperating hospital from a list of KPIs that ORTEC is currently using for another hospital:

- Distribution weekend shifts, which indicates how evenly these shifts are distributed between the nurses
- Long series of consecutive workdays, i.e. the number of times a nurse works a lot of days in a row
- Single rest days between workdays, which is the number of times a nurse has only one rest day between two working days
- Two consecutive weekends, which is the number of times a nurse works two weekends in a row

These KPIs are all formulated such that a lower number is better. The distribution of weekend shifts is the only one that can not be computed by simply counting the number of occurrences. For this KPI the same formula can be used that is used for the evening and night shifts in Section 3.4.2. The ML model should incorporate these KPIs into the scores by learning from the patterns in the previous schedules, which is why they are not included as soft constraints.

The schedules that are created using the methods in this section are, therefore, compared using the total score, the F1-score and the KPIs. Based on this comparison it is then evaluated whether the approach combining ML and OR is able to create good schedules.

To evaluate the usefulness of the alternative planning approaches, these criteria are still important. However, the decentralized and centralized cases are quite different and because of that the goal of the optimization also differs between the scenarios. For the scenario with centralized planning another goal of the approach is to examine whether the number of shifts

done by flex nurses could be lowered by scheduling differently or exchanging nurses between departments. For this reason the resulting schedule in this case is also evaluated on the number of shifts done by flex nurses, besides the criteria used for decentralized planning.

For the evaluation of the differences between the monthly and quarterly planning, the quality of the schedules is again used. This extension is included for two reasons, namely, to see how big of an improvement is made by the extra freedom it gives in satisfying the yearly labor laws and to evaluate the performance of the methods on larger instances. This is especially true for the centralized case as the quarterly planning could give an indication of how the methods will react to instances where even more departments are planned at the same time. The runtime of the methods for the quarterly case is, therefore, also interesting.

Furthermore, the metaheuristics are only run for the quarterly case due to time constraints and the fact that these methods are likely to be more useful for larger instances where the MIP is no longer able to solve the problem. For this reason, the quarterly case is the one that is mainly used to determine which of the methods works best. For the comparison of the different methods the quality of schedules is again evaluated using the described criteria and the runtime each model needs is also taken into account. However, this runtime should only be considered when schedules are nearly the same quality, since this is not the main concern of the planners as was mentioned previously.

## 5 Results

The results, obtained using the methods described in the previous section, are shown in this section. The methods were implemented in C# on .Net framework 4.8 and the MIP presented in Section 4.2 is modeled using Cplex version 22.1.1. In Section 5.1 the quality of the created schedule is evaluated for the decentralized monthly planning approach. Afterwards, the alternative planning approaches are evaluated in Section 5.2. Finally, a comparison of the different methods and their performance is given in Section 5.3.

### 5.1 Quality Created Schedule

In Table 2 the average score per assignment and the runtime needed are shown for the UB schedule, the simple heuristics, the MIP and the realized schedule. For the values in this table the average is taken over the three planned months. The score per assignment is computed by dividing the total score by the number of assignments that need to be made to create the schedule and it is shown instead of the total score due to the different instance sizes between departments and between months.

		RS	UB	H1	H2	MIP
IC	Score per assignment	0.6111	0.7005	0.6100	0.6244	0.6622
	Runtime (s)	-	-	0.01	0.01	42.22
Cardio	Score per assignment	0.4576	0.5944	0.5482	0.5440	0.5711
	Runtime (s)	-	-	0.00	0.00	1.98
Neuro	Score per assignment	0.4762	0.5603	0.5154	0.5068	0.5344
	Runtime (s)	-	-	0.00	0.00	6.73

Table 2: Comparison of the average ML-score per assignment and runtime needed in seconds when planning one month

Remark that the average score for UB is an upper bound on the score for the other schedules. In an ideal scenario, where the ML model predicts perfectly, this would be equal to the score for the realized schedule, but this is not the case for these instances. In the table it can be seen that making UB feasible using H1 results in a big decrease in score. However, it is still higher than the score for the realized schedule. The scores for H1 and H2 also seem to be quite close to each other, even though they start from different starting solutions. It should, however, be noted that these heuristics do not always give a feasible solution, which is caused by the fact that changes are made without considering long term consequences. This could result in cases where it is impossible to fill the duty capacity while maintaining feasibility due to the scheduled shifts on the surrounding days for the nurses that are skilled enough to fill this capacity.

The table also shows that the MIP comes quite close to the upper bound on the score and that, on average, it can be solved within a minute for all three departments. The runtime does seem to increase considerably for larger instances as, the average runtime for the IC department increases drastically compared to the two smaller departments.

The micro, macro and weighted F1-score are shown for the four schedules in Figure 5, with an average again being taken over the three instances for each department. As mentioned before, this F1-score should give an indication of how similar the resulting schedule is to the realized schedule.

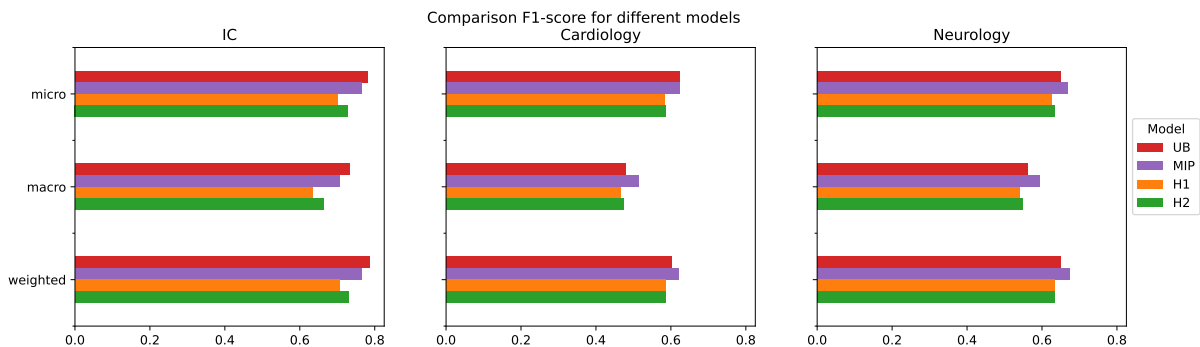


Figure 5: F1-scores w.r.t. the realized schedule for each model

The figure shows that the F1-score decreases when going from UB to H1, which is to be expected when looking at the decrease in score. It can also be seen that the F1-scores from the MIP and UB are nearly the same and that they are even slightly higher for the MIP. This could

be an indication that there are factors that could not be recognized by the ML model and that these missed factors are then included by the MIP. This would then correct some mispredictions and result in a higher F1-score.

These F1-scores do still seem to be pretty low, with the Cardiology and Neurology department having scores around 0.6, which means the schedules are not very close to the realized schedule. However, as mentioned before, the schedules can still be good even if they are quite different from the realized one. To determine the quality of these schedules, the KPIs can be computed and then compared to the one for the realized schedule, which is shown in Figure 6. In the figure the dashed blue line corresponds to the value of the KPIs in the realized schedule and all bars correspond to the percentual difference with this value. Here, bars left of the blue line show an improvement and bars right of the line show that the schedule performed worse on the KPI. Again the average is taken over the instances.

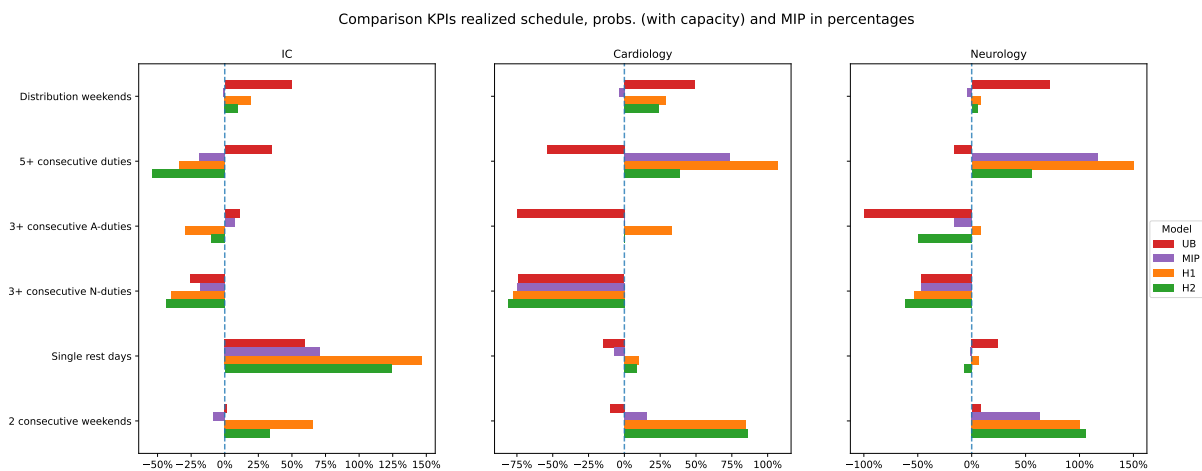


Figure 6: KPI comparison where all values are scaled as a percentual deviation from the realized schedule

The table shows that the schedule from the MIP performs quite well on the KPIs and that they are either close to realized ones or an improvement on most of them. However, it can be seen that removing the infeasibilities from the realized schedule does come at a cost, as one of the KPIs gets significantly worse for all departments. For the IC department this is the number of single rest days, while it is the KPI on more than five consecutive duties for both the Cardiology and the Neurology department.

## 5.2 Alternative Planning Approaches

In this section the alternative planning approaches are evaluated. First, the results for the centralized monthly planning approach are presented in Section 5.2.1 and compared to the results for the decentralized approach. Then the quarterly cases for both approaches are discussed in Section 5.2.2 and Section 5.2.3. For the results of the SA algorithm in these sections, the best found solution from several runs of the algorithm is presented. This schedule is chosen because the main objective in this section is to evaluate the planning approach, which can be done more fairly when using the best solution that can be found. However, a more in depth analysis of the different runs is given in Section 5.3, since the results in this section can give an overly

optimistic outlook on the performance of this method. For all runs the following parameters are used:  $T_{\max} = 10$ ,  $T_{\text{end}} = 0.5$ ,  $\alpha = 0.99$ , the maximum number of resets is set to 2 and the maximum number of iterations is 100,000. An example of the initial weights that are chosen and their progression during the algorithm can be found in Appendix A for one of the runs.

### 5.2.1 Centralized Monthly Planning

Table 3 again shows the average score per assignment and the runtime for several schedules. However, in this table the number of shifts done by flex nurses is shown as well. For the MIP the results are also shown for the decentralized case and the centralized one, to compare the two approaches. The flex nurses are excluded in the computation of the score per assignment in this table. Furthermore, the MIP is run with a time limit of four hours and the optimality gap is also shown in the table. This gap is the one reported by Cplex when reaching this time limit. All four values are again averaged over the three planned months.

	RS	UB	H1	H2	MIP	MIP-Decentral
Score per assignment	0.5383	0.6389	0.5567	0.5412	0.5811	0.6075
Flex shifts	426.33	652.33	289	211	138.33	426.33
Opt. gap (%)	-	-	-	-	1.951	-
Runtime (s)	-	-	0.06	0.07	14,413.45	152.81

Table 3: Comparison of ML-score per assignment, runtime needed in seconds, the number of duties done by flex nurses and the optimality gap reported by Cplex

In the table it can be seen that the score per assignment for the simple heuristics is a lot lower than the UB. These heuristics are also not guaranteed to result in a feasible schedule, as was the case with decentralized planning. Although shifts can be done by flex nurses and these flex nurses seem to be used less in the heuristics, there are still several days where the duty capacity can not be met without violating any constraints. This is due to the fact that there can be days where more than 20 flex nurses are needed and, while this can be solved by adding more flex nurses, this is not always possible in reality. Most of these days are during a weekend, as the added constraint on consecutive worked weekends appears to be especially tough to satisfy using these simple changes.

Furthermore, the table shows that the MIP is able to obtain a nearly optimal solution as the average optimality gap is around 2%. However, it was not able to obtain an optimal solution for any of the three months, which is why the average runtime is equal to four hours. It can also be seen that the MIP is able to decrease the number of flex shifts by 288 shifts compared to the decentralized MIP, which is a 67% decrease. This does, however, come with a 4.34% decrease in the score per assignment.

To see how close the generated schedules are to the realized schedule, Figure 7 shows the F1-scores w.r.t. the realized schedules.

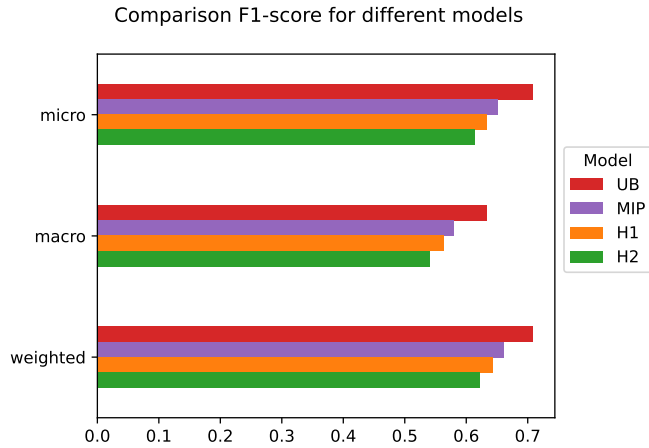


Figure 7: F1-scores w.r.t. the realized schedule for each model

In the figure it can be seen that the F1-scores for the MIP are a lot lower than the ones from UB, unlike in the decentralized case where the MIP even performed better for several instances. However, this could be a consequence of the decrease in the number of shifts done by flex nurses as these shifts need to be done by nurses within the department, which results in a lower F1-score. It can also be seen that the simple heuristics perform even worse than the MIP, even though they need more flex shifts and are infeasible.

Finally, a comparison of the KPIs to the realized schedule is presented in Figure 8. For this figure the KPIs are scaled in the same way as was done in the decentralized case. It should also be noted that the flex nurses are excluded in the calculation of these KPIs, as it is hard to measure the KPIs for flex nurses in the realized schedule and including them only in the generated methods would be unfair.

Comparison KPIs realized schedule, probs. (with capacity) and MIP in percentages

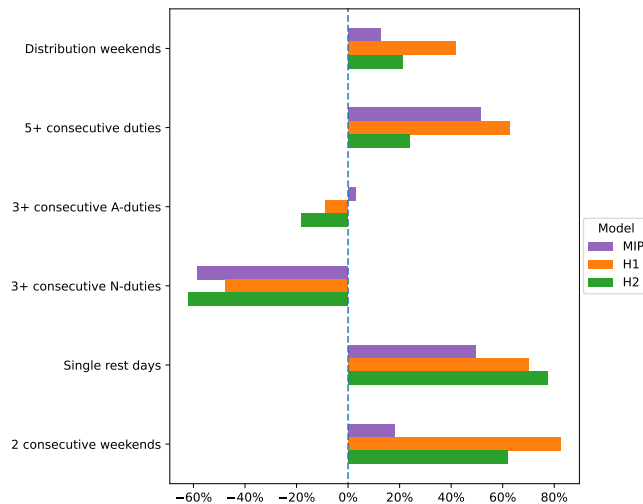


Figure 8: KPI comparison where all values are scaled as a percentual deviation from the realized schedule

The MIP again seems to perform worse than in the decentralized case, which is likely due to the decrease in flex shifts. However, the MIP still performs similarly to the realized schedule on four of the six KPIs, even with the 67% decrease in flex shifts.

## 5.2.2 Decentralized Quarterly Planning

For the decentralized quarterly planning approach, the average score per assignment and the runtime needed are computed for all methods. In Table 4 a comparison is made between the monthly and quarterly approach for the simple heuristics and the MIP. Note that there might be slight differences between the scores per assignment in this table and the ones in Table 2. This is due to the way the average is taken for these scores, as the scores in Table 2 are obtained by first computing the average score per assignment for each month and then taking the average of these averages for the three months. Therefore, the difference in length between the months is not taken into account, while this difference is taken into account when computing the scores in this table. For the metaheuristics only the quarterly case is presented because of the reasons mentioned in Section 4.4. The runtime for SA is also not completely known, since the best solution is chosen for multiple runs and most runs were done without keeping track of the runtime. For this reason, the runtime is shown for the single run that was timed, as a lower bound on the time needed for all runs.

		H1		H2		MIP		SA	VNS
		Monthly	Quarterly	Monthly	Quarterly	Monthly	Quarterly	Quarterly	Quarterly
IC	Score per assignment	0.6096	0.6001	0.6255	0.6192	0.6636	0.6680	0.6644	0.6565
	Understaffing	55	46	22	25	-	-	-	-
	Runtime (s)	0.03	0.05	0.03	0.04	126.66	4,992.30	>2,727.59	212.60
Cardio	Score per assignment	0.5477	0.5424	0.5438	0.5378	0.5710	0.5725	0.5716	0.5593
	Understaffing	24	20	5	11	-	-	-	-
	Runtime (s)	0.01	0.01	0.01	0.01	5.96	85.48	>3,537.28	95.22
Neuro	Score per assignment	0.5152	0.5105	0.5063	0.5064	0.5340	0.5367	0.5361	0.5218
	Understaffing	91	90	39	49	-	-	-	-
	Runtime (s)	0.01	0.01	0.01	0.01	20.19	135.78	>4,567.92	72.24

Table 4: Comparison of the ML-score per assignment and runtime needed in seconds when planning three months as well as the amount of understaffing in the schedule

The table shows that the average score per assignment decreases when creating a quarterly schedule for both H1 and H2. However, this decrease in score does come with a decrease in the number of times the duty capacity is not met for H1. But despite this, the heuristic is still not able to find a completely feasible schedule. Additionally, the table shows that the score per assignment increases by around 0.5% for the MIP when planning quarterly instead of monthly. But the runtime needed also increases significantly, with the MIP even taking more than an hour for the IC department.

It can also be seen that the score per assignment for the SA and VNS is quite close to the one obtained by the MIP. The schedule created using SA even has a slightly higher score than the one created using the MIP for the monthly case. However, the table also shows that the running time needed for both metaheuristics is significantly higher than the time needed to obtain the optimal solution using the MIP. Additionally, the use of SA is able to make the gap to the optimal solution smaller for all departments compared to the VNS. However, this improvement does come at a cost as the runtime needed increases drastically when using SA instead of VNS.

The F1-scores and KPIs for the quarterly approach do not change much from the monthly case, which is why they are not shown. However, they can be found in Appendix C.



### 5.2.3 Centralized Quarterly Planning

For the centralized quarterly planning approach the MIP is not able to find a feasible solution, even after several hours. It can, therefore, not be compared to the monthly planning. But this comparison can still be made for the simple heuristics, which is done in Table 5. In this table the results for the MIP are shown for the monthly case, which can be compared to the results for the metaheuristics. Since the total runtime is again not known for the multiple runs of the SA algorithm, the time needed for a single run is shown as a lower bound in this table as well.

	H1		H2		MIP	SA	VNS
	Monthly	Quarterly	Monthly	Quarterly	Monthly	Quarterly	Quarterly
Score per assignment	0.5557	0.5422	0.5410	0.5290	0.5815	0.5868	0.5786
Flex shifts	867	1,001	633	846	415	492	640
Understaffing	126	97	79	67	-	-	-
Runtime (s)	0.18	0.21	0.21	0.34	43,240.35	>13,112.771	7,307.042

Table 5: Comparison of ML-score per assignment, runtime needed in seconds and the number of duties done by flex nurses

It can be seen that the quarterly case of the heuristics seems to perform worse than the monthly case based on the score and the flex shifts. However, the quarterly approach is again able to produce a schedule that is closer to being completely feasible.

The table also shows that the score per assignment for the schedules created by SA with the quarterly approach is slightly higher than the score of monthly MIP. However, this increase in score does come at a cost as the number of flex shifts is increased by 77. Furthermore, it can again be seen that SA is able to outperform the VNS as it does better on both the score and flex shifts, but SA does likely take more than twice as long. In terms of runtime, using the metaheuristics with the quarterly approach also seems to outperform the monthly MIP. Although it is hard to evaluate the runtime for SA, the different runs can be run in parallel which means that the runtime should not increase by too much.

Since the F1-scores are again very similar to the ones for the monthly case, they can be found in Appendix C instead of this section. On the other hand, there are some differences in the KPIs between the monthly MIP and the metaheuristics. For this reason, they are shown in Figure 9 and they are again scaled to represent a percentual deviation from the realized schedule. In this figure the simple heuristics are left out since they did not change much from the monthly case.

Comparison KPIs realized schedule, probs. (with capacity) and MIP in percentages

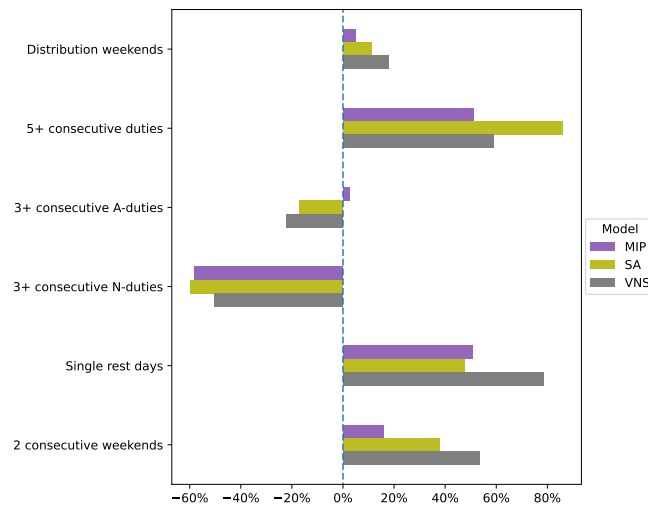


Figure 9: KPI comparison where all values are scaled as a percentual deviation from the realized schedule

The two metaheuristics seem to perform quite similarly on the KPIs, as differences on a given KPI seem to be balanced out by another KPI. Furthermore, the figure shows that the metaheuristics perform worse than the monthly MIP on most KPIs, although the difference does not appear to be that big. This means that, although the score can be improved by planning quarterly instead of monthly, this does cause a worse performance on the number of flex shifts and the KPIs.

### 5.3 Comparison Methods

A more in depth comparison of the methods is presented in this section. Since the simple heuristics are not able to satisfy the duty capacity, they are not included in this comparison, which means the focus is on the MIP and the two metaheuristics. First a small analysis is done on the effect the size of the MIP has on the time needed to find the first feasible solution. This is done to find a possible explanation for the struggles the MIP has with the centralized quarterly approach and can be found in Section 5.3.1. Afterwards, the SA algorithm is analyzed in more detail. As mentioned in the previous section, several runs are done due to the random elements in the algorithm and the differences between these runs are examined in Section 5.3.2. Finally, the methods are compared to each other for the different planning approaches in Section 5.3.3.

#### 5.3.1 Time Until First Feasible Solution MIP

Since the MIP struggled to find any feasible solution for the centralized quarterly approach, the time it took to find the first feasible solution and the number of variables in the model are analyzed for the other instances. Using these numbers, the scatter plots in Figure 10 can then be created, which should give an indication of the performance of the MIP for different instance sizes.

Comparison number of variables MIP and time needed to find feasible solution

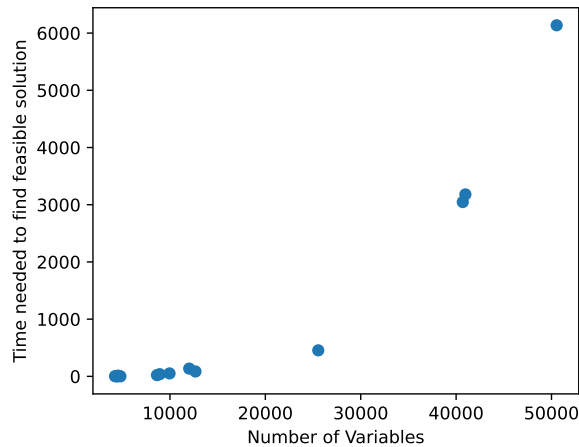


Figure 10: Comparison of the number of variables in the MIP and the time needed to find a feasible solution

The first thing that stands out from the figure is that there are very few large instances. However, from the points that are in the scatter plot, it seems that there is a superlinear relation, e.g. quadratic or exponential, between the number of variables in the MIP and the time it takes to find a feasible solution. This is to be expected since the NRP is known to be NP-hard, which means that finding any feasible solution to the problem is likely also NP-hard. This would also explain why the MIP struggles to find a feasible solution for the centralized quarterly planning approach, as the MIP has 130,799 variables for this instance, which is more than twice the number of variables of the largest instance that could be solved in this figure.

Another thing to note is that the MIP is also not able to improve on a feasible solution when providing a warm start for the centralized quarterly case. This shows that it is not just hard to find an initial feasible solution for such a large instance, but also to go from one feasible solution to another.

### 5.3.2 Analysis Runs SA

To examine how the solution develops during the SA algorithm and how the resulting schedule is created, the current and best found objectives is shown for each iteration in Figure 11, as well as the penalty that is incurred by the violations of constraints. In this figure only feasible solutions are considered for the best found objective, which is why the line only starts after a feasible solution is found. The figure shows one run for each of the three departments used in the decentralized approach and a run for the centralized instance.

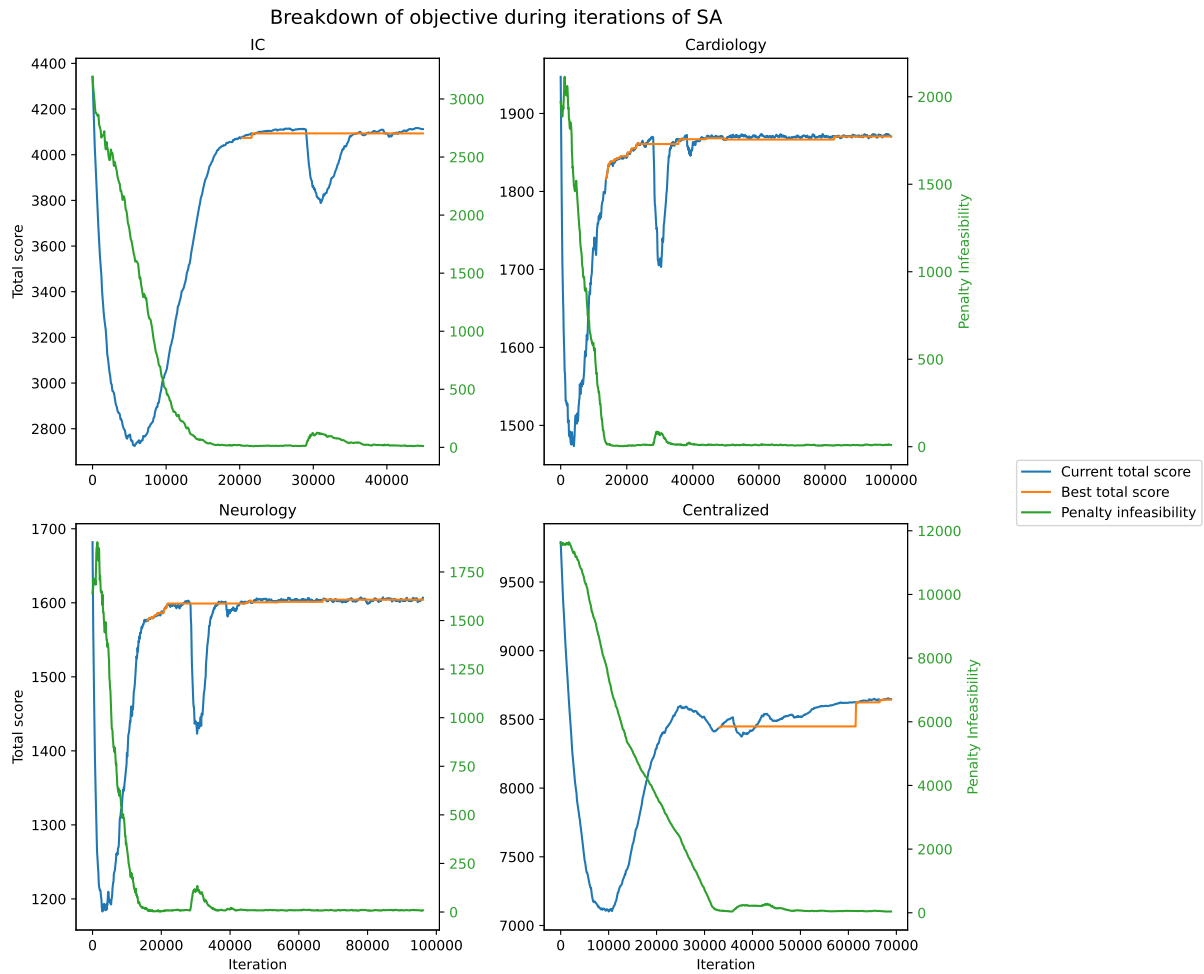


Figure 11: Total score of schedule for current and best found solution as well as penalty incurred by violations of constraints in current solution during iterations of SA

For all four instances a trade-off can be seen between the total score and the feasibility of the solution in the first stages of the algorithm, as the score is lowered to remove the violations. However, the figure also shows that the algorithm is able to improve the total score drastically after removing most violations. For the decentralized instances several sudden drops in the current total score can be seen, which correspond to the temperature resets and, while it may not be as clear for the centralized case, similar decreases can still be seen. From the increase in the best found total score it can be seen that these resets are able to improve the best found solution for three of the four instances. For the Cardiology and Neurology department this only seems to improve the solution slightly, but for the centralized case this improvement is a lot more evident.

For the Cardiology department there also seems to be a slight decrease in the best found objective. This could be due to the added case specific constraints, such as the number of partial weekends, for which a certain number of violations is allowed. Since a penalty is still incurred for a violation even if it is within this allowed number, a solution with a worse objective but with close to 0 of these violations could be considered better than one with the maximum allowed number of violations. It can also be seen that the penalty on violations still moves up and down slightly after finding a feasible solution. This shows that the weights on the penalties are

likely set properly as there is a balance between keeping the solution feasible and traversing the solution space freely by allowing some violations.

Another thing to note is that the first feasible solution for the centralized case is found after around 30 minutes, which is the point where the orange line starts. It can also be seen that the best found total score does not change often after this feasible solution is found. This might indicate that it is hard to improve the solution while moving from one feasible solution to another and that allowing some infeasibility is able to make this easier.

Due to the random elements that are included in the SA algorithm, several runs of the algorithm were done with different seeds. Table 6 shows the highest and the lowest optimality gap for the schedules that are obtained from 20 runs of the algorithm, as well as the range, mean and the standard deviation of the different runs. In order to make the results reproducible, the number from 1 to 20 are used as the random seeds for the runs. Since the optimal solution is not known for the centralized case, the gap is computed using an upper bound that is found by relaxing the quadratic constraints on the distribution of evening and night shifts and using the solution from SA as a warm start. A short analysis is done on the quality of this upper bound in Appendix B, which shows that it seems to be a very good bound.

	IC	Cardiology	Neurology	Centralized
Mean	0.933	0.347	0.432	3.486
Standard dev.	0.297	0.119	0.234	0.711
Min	0.544	0.155	0.110	2.478
Max	1.485	0.524	0.859	4.554

Table 6: Information on the optimality gap of the solutions of SA runs with different seeds for the centralized case the gap is calculated to an upper bound found by relaxing the quadratic constraints and solving the MIP

The table shows that the range increases quite drastically between departments as it is tripled when going from the Cardiology department to the larger IC department and then doubled when going from the IC department to the centralized approach. With this the standard deviation also doubles with every step, showing that the algorithm does indeed become less consistent for larger instances. However, the Neurology department seems closer to the IC department in terms of both the range and standard deviation, even though it is nearly the same size as the Cardiology department. This could be an indication that there are other factors that influence the consistency of the algorithm besides the size of the instance.

To give a more detailed insight into the different runs, Figure 12 contains histograms of the optimality gap for the schedules that are obtained using the different seeds. In this histogram bins of 0.05% are used for the decentralized instances while bins of 0.1% are used for the centralized instance.

Optimality gap for SA runs with different seeds

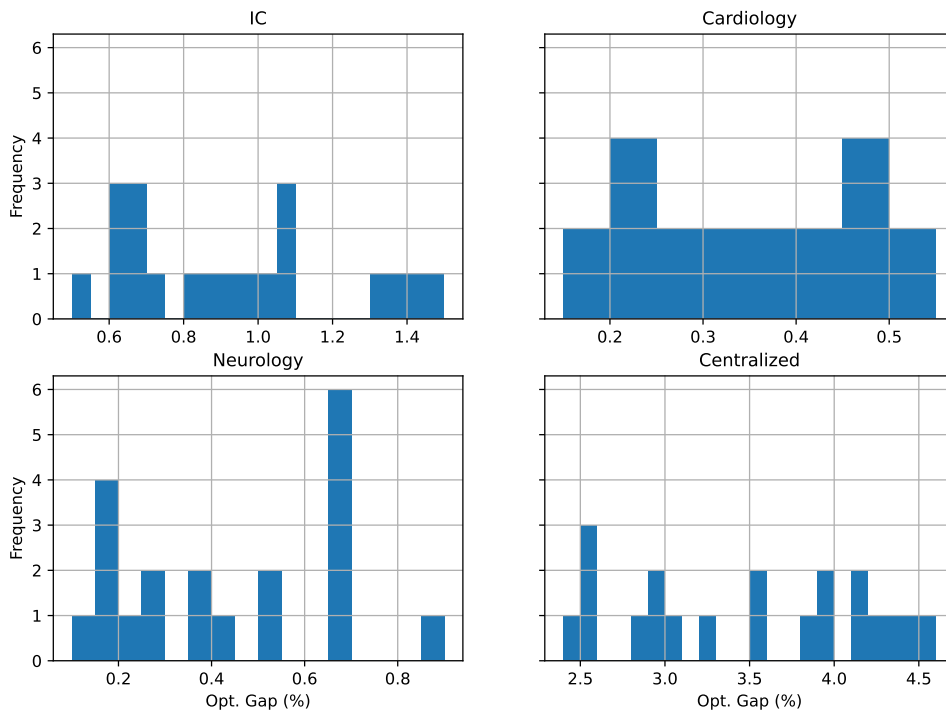


Figure 12: Histogram showing frequency of obtaining a solution with given optimality gap from SA with bins of width 0.05 for the decentralized departments and bins of width 0.1 for the centralized approach

The figure shows that there does not seem to be a clear bias towards the better or worse solutions, as the Cardiology and Neurology departments seem pretty balanced, the IC department seems to lean slightly towards the better solutions and the centralized approach might be slightly biased towards worse solutions. It can also be seen that there are no really big peaks, which shows that the gaps are spread quite evenly.

### 5.3.3 Comparison

Since the metaheuristics could not be run for the monthly approaches due to time constraints, the methods are mainly compared using the quarterly cases. Additionally, it is quite unlikely that the metaheuristics are able to outperform the MIP for the decentralized monthly case, since the optimal solution can be found within several minutes. Even if the metaheuristics would be able to find the same solution faster, this difference in time would likely not be very significant.

The score per assignment and runtime for the best found solutions of the SA and VNS are shown in Table 7 for the decentralized quarterly approach. The optimality gap is shown in the table as well, where the gap is calculated as the difference in total score between the optimal solution, found by the MIP, and the solution provided by the evaluated method. For SA the results are shown for the best of the 20 previously presented runs and a single run that is done with 0 as the random seed, independent of the other runs. This is done to show the difference in the results if the algorithm is run one time and if it is run multiple times with different seeds.

		MIP	SA		VNS
			Single run	Best run	
IC	Score per assignment	0.6680	0.6615	0.6644	0.6565
	Opt. gap (%)	-	0.975	0.544	1.726
	Runtime (s)	4,992.30	2,727.59	-	212.60
Cardio	Score per assignment	0.5725	0.5710	0.5716	0.5593
	Opt. gap (%)	-	0.271	0.155	2.313
	Runtime (s)	85.48	3,537.28	-	95.22
Neuro	Score per assignment	0.5367	0.5342	0.5361	0.5218
	Opt. gap (%)	-	0.470	0.110	2.775
	Runtime (s)	135.78	4,567.92	-	72.24

Table 7: Comparison of ML-score per assignment and runtime needed in seconds as well as the gap to the optimal solution

It can be seen that the score per assignment for the SA and VNS is quite close to the optimal one obtained by the MIP, as the gap between the two is below 3% for all departments. Additionally, it can be seen that SA is able to make the gap to the optimal solution smaller for all departments compared to the VNS regardless of whether a single run or multiple runs are done. Doing multiple runs of SA also improves the solution quite a bit as the gap is nearly halved for all departments.

In terms of runtime the MIP and VNS seem to perform equally well for the two smaller departments, as both are able to find a schedule within a couple of minutes, while the SA algorithm needs more than an hour for these instances. However, the MIP needs nearly 90 minutes to find the optimal solution for the larger IC department. On the other hand, VNS is able to find a solution that is quite close to the optimal one within 4 minutes and an even better solution can be found using a single run of SA, although this does require around 45 minutes. Additionally, if multiple runs of SA can be done in parallel, the runtime should not increase too drastically and a solution that is only around 0.5% worse than the optimal solution can be found.

For the centralized quarterly case, Table 8 shows the score per assignment, the number of flex shifts and runtime. In the table these values are shown for the metaheuristics and the schedule that is obtained by connecting the monthly schedules obtained using the MIP for the monthly case. The table also contains the optimality gap to the upper bound described in the previous section and the results for SA are again shown for both a single run and the best run.

	MIP	SA		VNS
	Monthly	Single run	Best run	
Score per assignment	0.5815	0.5789	0.5868	0.5786
Flex shifts	415	587	492	640
Opt. gap (%)	2.351	4.677	2.478	5.296
Runtime (s)	43,240.35	13,112.77	-	7,307.04

Table 8: Comparison of the objective, score per assignment, the number of shifts done by flex nurses, runtime needed in seconds and optimality gap to an upper bound found by relaxing the quadratic constraints and solving the MIP

The table shows that the schedule that is created by connecting the monthly schedules from the MIP has the best score per assignment and the lowest number of flex shifts. Doing multiple runs of SA results in the second best performance on these objectives, as the score is even slightly higher for this schedule but more flex shifts are needed which results in the optimality gap being slightly bigger. It can also be seen that doing a single run of SA is already able to outperform the VNS as it does better on both the score and flex shifts, but the difference is quite small.

Although the MIP is able to produce the best schedule, it does have a runtime of around 12 hours, while the VNS takes 2 hours and a single run of SA about 4 hours. Additionally, multiple runs of SA can again be done in parallel, which should not increase run time too much and can result in a schedule that is a lot closer in quality to the one created by the MIP. However, another factor that should be taken into account in this comparison is the fact that the KPIs for the metaheuristics are slightly worse, which was shown in Section 5.2.3.

## 6 Discussion

Based on the results shown in Section 5, it seems that the planning of nurses using a combination of ML and OR works quite well. It is also shown that only ML or a combination of ML and a simple heuristic is not enough to create a schedule, but that either a MIP or a metaheuristic is needed. By encapsulating soft constraints into the scores that are predicted by the ML model, it becomes possible to formulate the problem with only one term in the objective instead of a weighted sum, which is most commonly used in literature. The problem is also simplified by excluding the soft constraints, which allows for the addition of quite complex constraints, such as the quadratic constraints on the distribution of shifts between nurses. Due to these changes, the MIP also seems to perform quite well and it is able to provide schedules that are similar in quality to the realized schedule based on the KPIs the cooperating hospital has provided, although it is not able to create schedules that are similar to the realized one. It is also able to create a schedule with a significantly higher total score than the realized schedule, which implies that the found schedule is better than the realized one according to the scores. However, the MIP does seem to struggle with very large instances. This can be seen with the centralized quarterly planning approach, where the MIP is not able to find a feasible solution or improve on a given feasible solution due to the large number of variables. An explanation for these struggles could be that there seems to be a superlinear correlation between the number of variables in the MIP and the time it needs to find a feasible solution.



Additionally, planning three months at a time instead of one month does not seem to improve the schedules by much, even though there is more freedom in how the yearly labor laws are satisfied. This means that it is likely not worth the extra effort that the nurses and planners need to put in to create these quarterly schedules. On the other, the centralized planning approach is able to reduce the number of shifts that need to be done by flex nurses significantly. However, this does come at a cost, as the KPIs and the total score get worse. Since the evaluation of this trade-off between the use of flex nurses and the quality of the schedule can differ heavily between hospitals, and even between different planners, the results regarding these findings are left open for interpretation.

From the results it can also be seen that the SA algorithm is able to find nearly optimal solution for smaller instances as a schedule can be found that is at most 1% worse than the optimal solution for all departments, but it does take a long time. For larger instances the metaheuristic is still able to find reasonably good solutions as the total score of the resulting schedule is higher than the score of the realized schedule and the optimality gap is 3% if multiple runs are done with different seeds. Additionally, SA is shown to outperform VNS for all instances, although it does take more time. It should also be noted that SA is able to find an initial feasible solution quite fast for larger instances, as it is able to find a feasible solution in 30 minutes while the MIP is not able to find any feasible solution even after more than 5 hours. This initial feasible solution also has a fairly good objective, which could be useful if only a limited amount of time is available to run the metaheuristic. Furthermore, the SA algorithm is quite consistent for different random seeds for smaller instances, which means that running it once is likely enough since other runs will give comparable results. However, for larger instances, especially the centralized cases, SA becomes less consistent and it is definitely useful to do several runs with different seeds as it could result in quite large improvements. Adjusting the weights during the algorithm is also very helpful as less time needs to be spent on the tuning of the initial weights. It also helps in finding a balance between the feasibility of the schedule and flexibility when traversing the solution space, since this balance can require different weights in different areas of the solution space.

The assumption that the realized schedule is good and the assumption that the ML model is able to predict perfectly are both unrealistic and these inputs are only used since there is no alternative. However, if a good schedule, e.g. the planned schedule, would be available and an ML model would be able to create perfectly predicted scores based on this good schedule, the MIP and SA could still be applied to these inputs without any changes. Because of these methods it might not even be necessary that the predictions from the ML model are perfect, since it could be seen that some mistakes in the ML model could be fixed by the MIP, as was shown by the increase in F1-score when using the MIP for the decentralized monthly planning approach.

The assumption that are made regarding the borrowing of nurses might also not be completely realistic, as the hospital indicated that the head nurse at the department usually indicates which nurses can be borrowed and which skill level they can do in another department. This should, therefore, differ between nurses, but because this data is not stored, the assumption needed to be made that all nurses could be borrowed and that they would be one skill level

lower in the other department. However, this could also be fixed relatively easily as long as the data is available.

It should also be noted that, due to the limited number of instances, it is hard to verify whether these results are widely applicable or that they are very specific to the instances that are available. This is also why the number of runs with different seeds in the evaluation of the robustness of the SA is kept quite low, as there is no real way of verifying whether the results can be generalized or that they rely on the fact that the instances just happen to be very easy or very hard to solve.

## 7 Conclusion

This thesis examines whether it is possible to plan nurses using a combination of ML and OR. To do this ORTEC developed an ML model that predicts scores that indicate how good the assignment of a nurse to a duty on a given day is. This score is then used as the objective in a MIP and a SA algorithm, where the total score of all assignments should be maximized. Using these methods several different planning approaches are also evaluated to determine whether it could be interesting to for the hospital to change their approach.

These methods were tested on data from one of the hospitals that ORTEC is working with, who provided data on three departments. Using this data the months of April, May and June of 2022 were planned, which showed that the combination of ML and OR seems to work. The methods are able to create schedules that outperform the realized schedule based on the predicted scores and perform equally well as the realized schedule on several KPIs. It could also be seen that some imperfections in the predictions of the ML model could be fixed by applying the OR methods, which implies that it might not be necessary to have a completely perfect ML model.

For most instances the MIP can be used to create this schedule and is able to provide optimal solution fairly quickly. However, it is not possible to solve very large instances using the MIP. To create a schedule for these large instances the SA algorithm can be used, which was shown to produce schedules with an optimality gap that is smaller than 5%. However, it does take quite a long time as an hour is needed for each of the smaller instances and around three hours is needed for the larger instances. If multiple runs can be done, the resulting schedule can be improved further as the gap becomes smaller than 3% and the runtime should not increase too drastically, as long as they can be done in parallel. Additionally, an initial feasible solution is found relatively quickly, which could be used as a warm start in a MIP or in another heuristic that might be faster.

It could also be seen that the current planning approach the hospital is using, which is to plan each department independently on a monthly basis, seems to be the best approach to use. Since planning on a quarterly basis only gives a small improvement over the monthly planning, it is likely not worth the extra effort that nurses and planners need to put in to make this possible. However, planning multiple departments simultaneously does make it possible to reduce the usage of flex nurses drastically, although the quality of the schedule does become worse. It could, therefore, differ between hospitals if this trade-off is worth it or not.

These results show the potential of solution methods that combine ML and OR to solve the NRP by predicting how good an assignment would be and then using these predictions

to create a schedule. This has not been investigated in any of the found literature, although a similar approach has been applied to the Crew Pairing Problem. Additionally, the SA algorithm, developed in this thesis, seems to be the first time that a heuristic approach is used as the OR side of the method for scheduling problems, as others focused on using MIPs and making these models solvable using, for example, column generation.

These findings do, however, rely quite heavily on the assumption that the ML model is able to predict scores that are a reasonably accurate representation of the reality, which might be unrealistic. Even if the OR methods are able to fix some minor imperfections, larger inaccuracies will still carry over into the resulting schedule.

Furthermore, it could be interesting to see whether the MIP could be made solvable for larger instances by applying column generation, because of how well the MIP seems to perform on the small instances and because this also done for the Crew Pairing Problem by Tahir et al. (2021) and Yaakoubi et al. (2020). This would eliminate the need to use metaheuristics as all instances could then be solved with the MIP.

An alternative to this would be to investigate ways to improve the SA algorithm. This could be done by speeding up the exploration of neighborhoods by evaluating several neighbors at the same time using parallel processing. This was not done in this thesis because the cooperating hospital indicated that the runtime was not very important, but it could still be useful for hospitals that do need a solution quickly. The algorithm could also be improved by applying hyper parameter tuning on the parameters, such as the start and end temperature, the number of resets and the cooling factor. Due to time constraints this was not possible in this thesis and instead only a small number of alternatives was tested with the best ones being chosen as the ones used. Other algorithms might also be interesting to investigate, such as evolutionary algorithms that are often used for the NRP in traditional approaches where no ML is used. The allowing of infeasibilities and adjusting of weights for the penalty terms for these infeasibilities could still be used in these other algorithms as this showed promising results with SA.

Alternatively, research could be done into changes in the ML model based on the results in this thesis. The ML model currently uses several features that are influenced by the nurses schedule for the previous day or week. For these features a rolling window approach is used where nurses are assigned the duty with the highest predicted score when determining the value of these features. However, the results showed that this method of assigning duties to nurses is flawed and it could, therefore, be interesting to use the MIP to assign the duties for this rolling window, especially since the MIP is able to solve small instances within seconds.

## References

- U. Aickelin, E. K. Burke, and J. Li. An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering. *Journal of the Operational Research Society*, 58(12):1574–1585, 2007.
- M. A. Awadallah, A. L. Bolaji, and M. A. Al-Betar. A hybrid artificial bee colony for a nurse rostering problem. *Applied Soft Computing*, 35:726–739, 2015.
- J. Baeklund. Nurse rostering at a danish ward. *Annals of Operations Research*, 222:107–123, 2014.
- F. Bellanti, G. Carello, F. Della Croce, and R. Tadei. A greedy-based neighborhood search approach to a nurse rostering problem. *European Journal of Operational Research*, 153(1):28–40, 2004.
- Y. Bengio, A. Lodi, and A. Prouvost. Machine learning for combinatorial optimization: A methodological tour d’horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.
- T. Breugem, T. Dollevoet, and D. Huisman. Is equality always desirable? analyzing the trade-off between fairness and attractiveness in crew rostering. *Management Science*, 68(4):2619–2641, 2022.
- E. Burke, P. De Causmaecker, and G. Vanden Berghe. A hybrid tabu search algorithm for the nurse rostering problem. *Lecture Notes in Artificial Intelligence*, 1585:187–194, 11 1998.
- E. Burke, P. Cowling, P. De Causmaecker, and G. Vanden Berghe. A memetic approach to the nurse rostering problem. *Applied intelligence*, 15:199–214, 2001a.
- E. Burke, P. De Causmaecker, S. Petrovic, and G. Vanden Berghe. Fitness evaluation for nurse scheduling problems. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, volume 2, pages 1139–1146, 2001b.
- E. Burke, P. De Causmaecker, S. Petrovic, and G. Vanden Berghe. *Variable Neighborhood Search for Nurse Rostering Problems*, pages 153–172. Springer US, Boston, MA, 2004a.
- E. Burke, P. De Causmaecker, G. Vanden Berghe, and H. Van Landeghem. The state of the art of nurse rostering. *J. Scheduling*, 7:441–499, 11 2004b.
- E. K. Burke, T. Curtois, G. Post, R. Qu, and B. Veltman. A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. *European Journal of Operational Research*, 188(2):330–341, 2008.
- S. Ceschia, L. Di Gaspero, V. Mazzaracchio, G. Policante, and A. Schaerf. Solving a real-world nurse rostering problem by simulated annealing. *Operations Research for Health Care*, 36:100379, 2023.
- B. Cheang, H. Li, A. Lim, and B. Rodrigues. Nurse rostering problems—a bibliographic survey. *European Journal of Operational Research*, 151(3):447–460, 2003.
- N. Cissen. Predicting rosters in healthcare by using machine learning. MSc Thesis Tilburg University, 2022.
- T. Cooper and J. Kingston. The complexity of timetable construction problems. In E. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling*, pages 281–295. Springer Berlin Heidelberg, 1996.

- P. De Causmaecker and G. Vanden Berghe. A categorisation of nurse rostering problems. *Journal of Scheduling*, 14:3–16, 2011.
- R. G. Drake. The ‘robust’ roster: Exploring the nurse rostering process. *Journal of advanced nursing*, 70(9):2095–2106, 2014.
- A. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3–27, 2004. Timetabling and Rostering.
- J. Fitzpatrick, D. Ajwani, and P. Carroll. Learning to sparsify travelling salesman problem instances. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 18th International Conference, CPAIOR 2021, Vienna, Austria, July 5–8, 2021, Proceedings 18*, pages 410–426. Springer, 2021.
- M. D. Goodman, K. A. Dowsland, and J. M. Thompson. A grasp-knapsack hybrid for a nurse-scheduling problem. *Journal of Heuristics*, 15(4), 2009.
- M. Hadwan. Annealing harmony search algorithm to solve the nurse rostering problem. *Computers, Materials and Continua*, pages 5545–5559, 2022.
- F. He and R. Qu. A constraint programming based column generation approach to nurse rostering problems. *Computers & Operations Research*, 39(12):3331–3343, 2012.
- Ö. Kelemci and S. Uyar. Application of a genetic algorithm to a real world nurse rostering problem instance. In *International Conference on Enterprise Information Systems (ICEIS)*, pages 474–477, 2007.
- D. L. Kellogg and S. Walczak. Nurse scheduling: From academia to implementation or not? *Interfaces*, 37(4):355–369, 2007.
- S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science (New York, N. Y.)*, 220:671–80, 06 1983.
- M. Kruber, M. E. Lübbecke, and A. Parmentier. Learning when to use a decomposition. In *Integration of AI and OR Techniques in Constraint Programming: 14th International Conference, CPAIOR 2017, Padua, Italy, June 5-8, 2017, Proceedings 14*, pages 202–210. Springer, 2017.
- M. Kumar, S. Teso, P. De Causmaecker, and L. De Raedt. Automating personnel rostering by learning constraints using tensors. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 697–704. IEEE, 2019.
- A. Lodi, L. Mossina, and E. Rachelson. Learning to handle parameter perturbations in combinatorial optimization: an application to facility location. *EURO Journal on Transportation and Logistics*, 9(4):100023, 2020.
- ORTEC. An advanced solution for work schedules. [https://ortec.com/assets/content/media/downloadable\\_content/ORTEC%20Workforce%20Scheduling%20-%20Brochure.pdf](https://ortec.com/assets/content/media/downloadable_content/ORTEC%20Workforce%20Scheduling%20-%20Brochure.pdf), 2021. Accessed: 2023-03-24.
- ORTEC. Ortec-about us. <https://ortec.com/en/about-us>, 2023. Accessed: 2023-03-24.
- I. Osman. Heuristics for the generalised assignment problem: simulated annealing and tabu search approaches. *Operations-Research-Spektrum*, 17:211—225, 1995.

- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.
- D. Quak. A novel hybrid machine learning metaheuristic approach to create nurse rosters in a dutch hospital. MSc Thesis University of Twente, 2023.
- A. Tahir, F. Quesnel, G. Desaulniers, I. El Hallaoui, and Y. Yaakoubi. An improved integral column generation algorithm using machine learning for aircrew pairing. *Transportation Science*, 55(6):1411–1429, 2021.
- D. Tayebi, S. Ray, and D. Ajwani. Learning to prune instances of k-median and related problems. In *2022 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 184–194. SIAM, 2022.
- G. M. Thompson. A simulated-annealing heuristic for shift scheduling using non-continuously available employees. *Computers & Operations Research*, 23(3):275–288, 1996.
- A. M. Turhan and B. Bilgen. A hybrid fix-and-optimize and simulated annealing approaches for nurse rostering problem. *Computers & Industrial Engineering*, 145:106531, 2020.
- J. Van den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester, and L. De Boeck. Personnel scheduling: A literature review. *European Journal of Operational Research*, 226(3):367–385, 2013.
- T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60:611–624, 06 2012.
- O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.
- J.-j. Wu, Y. Lin, Z.-h. Zhan, W.-n. Chen, Y.-b. Lin, and J.-y. Chen. An ant colony optimization approach for nurse rostering problem. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1672–1676. IEEE, 2013.
- Y. Yaakoubi, F. Soumis, and S. Lacoste-Julien. Machine learning in airline crew pairing to construct initial clusters for dynamic constraint aggregation. *EURO Journal on Transportation and Logistics*, 9(4):100020, 2020.

## Appendix A Weights SA

Figure 13 shows the weights for each of the penalty terms corresponding to a violation of one of the constraints. The progression of the weight during the iterations of the SA algorithm is shown for the three departments and the centralized approach.



Figure 13: Weights for the penalty terms in objective corresponding to violations during iterations of the SA algorithm

The figure shows that the weights are decreased drastically after a feasible solution is found. This shows that keeping the weights the same would be very restrictive during these iterations, which could make it quite hard to traverse the solution space.

## Appendix B Evaluation Upper Bound

To evaluate the upper bound that is used to determine the optimality gap for the centralized quarterly case, the same relaxation is performed for each department. The results of that relaxation are shown in Table 9, together with how heavily the relaxed constraints are violated.

	IC	Cardio	Neuro	Central
Total score	4133.619	1875.575	1611.809	-
Total score relaxed	4140.614	1875.901	1613.890	8863.279
Gap (%)	0.169	0.017	0.13	-
Violation evening distr. (%)	17.71	7.26	5.35	7.67
Violation night distr. (%)	6.03	0.00	13.54	0.00

Table 9: Comparison of total score, runtime needed in seconds and the number of duties done by flex nurses

The table shows that the relaxation seems to be a very strong upper bound as the gap is very small for all departments. It can also be seen that the Cardiology department and the centralized approach have a similar degree of violation, which is good since the department has the smallest gap between the regular and relaxed models.



# Appendix C Additional Results

## Decentralized Quarterly Approach

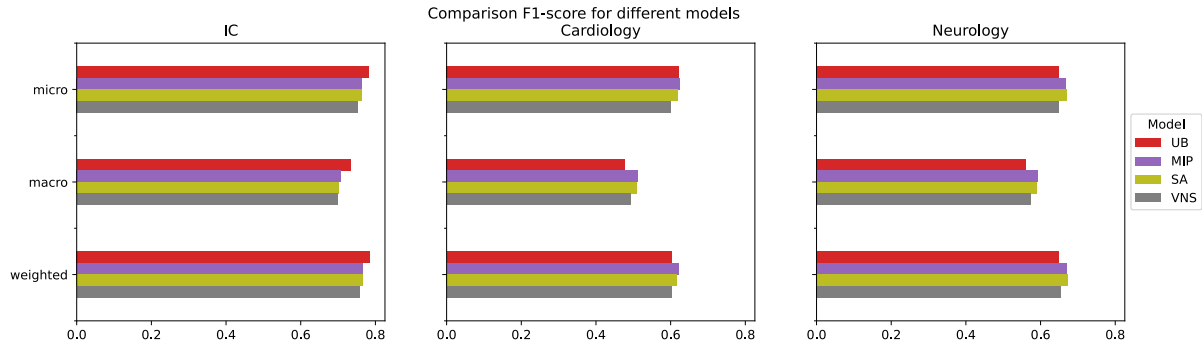


Figure 14: F1-scores w.r.t. the realized schedule for each model

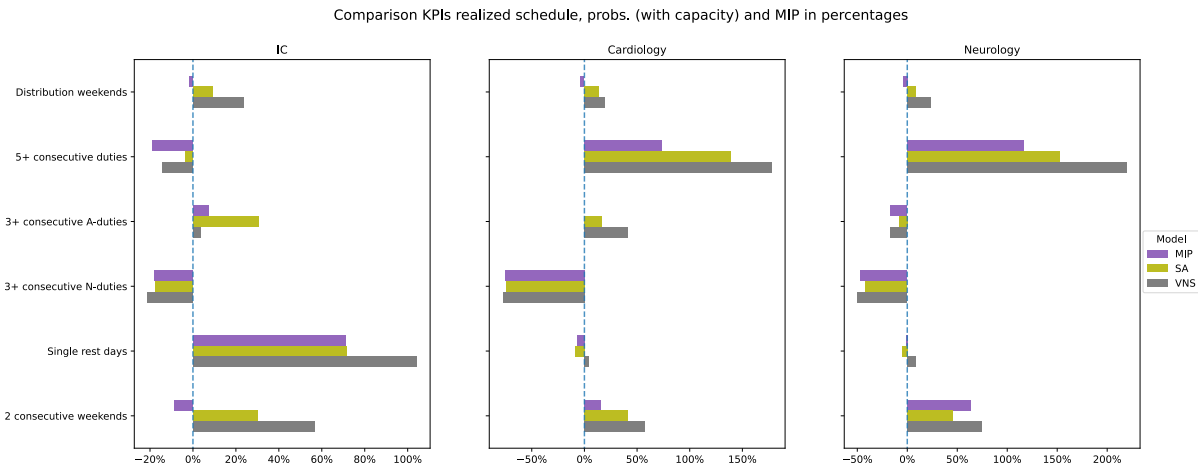


Figure 15: KPI comparison where all values are scaled as a percentual deviation from the realized schedule

## Centralized Quarterly Approach

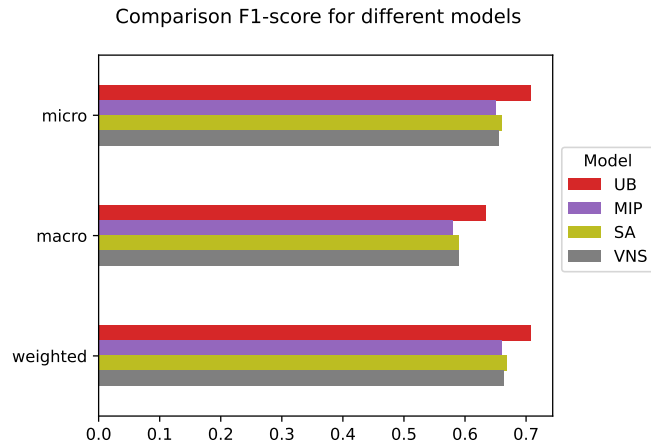


Figure 16: F1-scores w.r.t. the realized schedule for each model

Number of duties done in the realized schedule, the probs. (with capacity and CAO) and the MIP

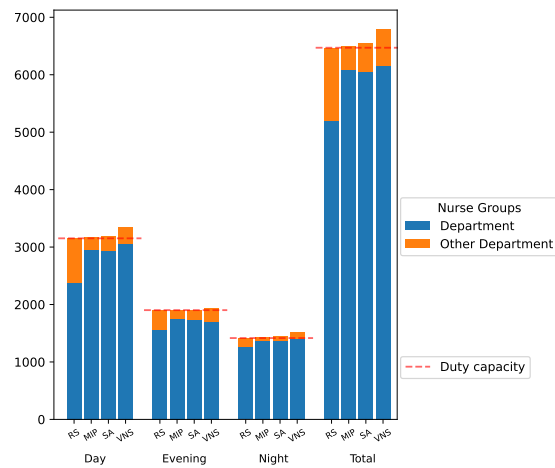


Figure 17: Number of shifts done by nurses within and outside of the department for each model

## SA Runs

seed	IC		Cardiology		Neurology		Centralized		
	Total score	Gap (%)	Total score	Gap (%)	Total score	Gap (%)	Total score	Flex shifts	Gap (%)
1	4,095.464	0.923	1,871.144	0.236	1,607.714	0.254	7,207.633	568	4.554
2	4,111.146	0.544	1,870.151	0.289	1,609.337	0.153	7,297.037	505	2.835
3	4,108.728	0.602	1,871.463	0.219	1,609.100	0.168	7,262.885	534	3.547
4	4,089.304	1.072	1,871.464	0.219	1,608.184	0.225	7,258.916	562	3.908
5	4,106.928	0.646	1,871.559	0.214	1,609.057	0.171	7,225.525	547	4.115
6	4,108.101	0.617	1,869.276	0.336	1,609.014	0.173	7,297.807	519	2.984
7	4,105.121	0.689	1,872.201	0.180	1,610.033	0.110	7,232.403	563	4.218
8	4,098.920	0.839	1,872.675	0.155	1,607.520	0.266	7,254.666	560	3.933
9	4,091.802	1.012	1,870.786	0.255	1,606.137	0.352	7,282.598	530	3.280
10	4,104.784	0.698	1,869.571	0.320	1,601.169	0.660	7,252.120	573	4.109
11	4,089.667	1.063	1,866.917	0.462	1,600.585	0.696	7,299.255	528	3.069
12	4,072.228	1.485	1,867.762	0.417	1,600.813	0.682	7,242.386	542	3.869
13	4,106.028	0.667	1,866.122	0.504	1,603.537	0.513	7,259.221	528	3.521
14	4,093.053	0.981	1,865.749	0.524	1,597.961	0.859	7,315.670	492	2.478
15	4,089.887	1.058	1,867.041	0.455	1,601.008	0.670	7,305.725	486	2.522
16	4,076.455	1.383	1,868.177	0.394	1,601.170	0.660	7,314.939	495	2.520
17	4,103.359	0.732	1,867.403	0.436	1,601.115	0.663	7,230.514	574	4.364
18	4,078.175	1.341	1,868.458	0.379	1,604.733	0.439	7,308.134	489	2.529
19	4,073.956	1.443	1,866.909	0.462	1,603.167	0.536	7,295.819	514	2.950
20	4,098.339	0.854	1,866.526	0.482	1,605.415	0.397	7,207.189	555	4.412

Table 10: Results for different seeds of SA