FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Exact and Heuristic Approaches for a Multi-Stage Nurse Rostering Problem

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Computational Intelligence

eingereicht von

Florian Mischek
Matrikelnummer 1025898

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: Priv.Doz. Dr. Nysret Musliu

Wien, 17. August 2016

Florian Mischek	Nysret Musliu

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Exact and Heuristic Approaches for a Multi-Stage Nurse Rostering Problem

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Computational Intelligence

by

Florian Mischek
Registration Number 1025898

to the Faculty of Informatics
at the TU Wien
Advisor: Priv.Doz. Dr. Nysret Musliu

Vienna, 17th August, 2016

Florian Mischek	Nysret Musliu

Erklärung zur Verfassung der Arbeit

Florian Mischek
Scheunenstr. 31, 2100 Korneuburg

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 17. August 2016

Florian Mischek

Acknowledgements

First, I want to express my gratitude to my advisor Priv.Doz. Dr. Nysret Musliu, for his invaluable support and guidance, as well as for his enthusiasm towards and commitment to this thesis and my work.

My thanks also go to my colleague Felix Winter, for always having an open ear and helpful advice for any complications and problems I faced during the creation of this thesis.

I cannot overstate my deep gratefulness for my family, who have been incredibly supportive, motivating and patient throughout all my life.

Finally, I also want to thank whoever is reading this, for taking an interest and being dedicated enough to read even this part.

This work was supported by the Austrian Science Fund (FWF): P24814-N23.

Kurzfassung

Die automatische Erstellung von Schichtplänen für Angestellte in Spitälern und ähnlichen Einrichtungen ist seit Jahrzehnten ein wichtiges Forschungsfeld. Der Bedarf an Arbeitskräften muss für jede Zeitperiode sichergestellt werden, während gleichzeitig gesetzliche und vertragliche Bestimmungen eingehalten und Wünsche der Angestellten berücksichtigt werden müssen.

In der für die Second International Nurse Rostering Competition (INRC-II) verwendeten Problemstellung müssen Schichtpläne für mehrere aufeinanderfolgende und untereinander abhängige Zeiträume erstellt werden. In diesem Format sind Informationen über die Anforderungen zukünftiger Planungsperioden erst verfügbar, nachdem der aktuelle Plan fixiert wurde. Das entspricht eher der täglichen Praxis, wie sie in Spitälern zu finden ist, wo sich die Anforderungen innerhalb kurzer Zeiträume ändern können, als die monolithischen Problemstellungen, die in der Literatur üblicherweise behandelt werden.

Diese Struktur macht es notwendig, dass Planungsprogramme bereits im Voraus zukünftige Planungsperioden mit unbekannten Anforderungen in ihre Lösungen miteinbeziehen, um Ungleichgewichte und hohe Einbußen bei der globalen Qualität der Lösungen zu vermeiden.

In dieser Diplomarbeit wird eine Integer Programming Formulierung des ursprünglichen Problems vorgestellt und dann mittels zusätzlicher Constraints erweitert, die diese speziellen Anforderungen berücksichtigen. Diese Constraints sind so allgemein formuliert, dass sie einfach auch an andere Probleme mit der selben Struktur angepasst werden können. Mithilfe dieser Erweiterungen werden deutliche Verbesserungen in der Qualität der erzeugten Schichtpläne erzielt, ohne dabei die von der INRC-II vorgeschriebenen Zeitlimits zu überschreiten. Außerdem wird ein heuristisches Local Search Framework implementiert, das sich ebenfalls die zusätzlichen Constraints zu Nutze macht, um seine Ergebnisse zu verbessern.

Die Resultate beider Methoden werden experimentell ausgewertet und es wird gezeigt, dass sie mit den Ergebnissen der Finalisten der INRC-II vergleichbar sind.

Abstract

The generation of working schedules for employees in hospitals and similar institutions has been an important field of study for multiple decades. Demand for each time period must be met, while at the same time conforming to various regulations, employment contracts and employee preferences.

In the problem variant posed for the Second International Nurse Rostering Competition (INRC-II), solvers need to produce schedules for multiple consecutive and interdependent periods. In this setting, also called a stepping horizon, information about future periods becomes available only after the current schedule has been fixed. This corresponds more closely to the real-word practice found in hospitals, where demand can change within short periods, compared to the monolithic problem formulations usually studied in the literature.

The problem structure requires solvers to take future stages with unknown requirements into account during the solution of the current week in order to avoid imbalances and high overall penalties to the quality of the whole schedule.

In this thesis, an Integer Programming model of the problem is introduced and then extended with additional constraints that consider the special requirements of this problem, but are general enough to fit also other problems of a similar structure. It is shown that using these extensions, substantial improvements in the quality of the generated schedules can be achieved, while still keeping solution times within the time limits imposed by the competition. Further, a Local Search framework is implemented that also makes use of these additional constraints to improve its solutions.

The results of both approaches are experimentally evaluated and compared to each other. They are shown to be competitive with those of the finalists in the INRC-II.

Contents

CHAPTER 1

Introduction

Generating high-quality working schedules is a complex, but very important problem in many areas of work. On the one hand, there are numerous legal requirements and labour regulations that have to be fulfilled. On the other hand, the quality of an employees schedule can have a major impact on their performance, health and satisfaction with their work (e.g. [KG89]). All this has to be balanced by the need to have enough employees present at all times to successfully cover the workload.

Generating such schedules manually requires lots of experience and is a very time consuming process. In addition, the schedules mostly are far from optimal concerning the criteria listed above. Consequently, for multiple decades, a large amount of research has been devoted to solving this problem in its numerous variants using automatic tools.

The exact tasks to be solved differ greatly between industries and even for different companies within the same field, making a general solution for all variants of the problem almost impossible. The requirements range across determining the demand, designing acceptable shifts, generating working patterns for employees and assigning individual tasks to each timeslot during work.

One area that is particularly challenging is the scheduling of nurses in hospitals. Care has to be provided around the clock, during all days of the week, while the demand varies widely within short timeframes. This problem is usually called *nurse rostering* or *nurse scheduling*. Different definitions of the problem exist in the literature, Burke et al. [BCBL04] defined nurse rostering as ”the allocation of nurses to periods of work over several weeks“.

Over the years, a large number of solution methods have been used to tackle problems of this kind, both exact (e.g. mathematical programming) and heuristic. Multiple surveys cataloguing and reviewing these publications exist, e.g. [BCBL04, EJKS04, dBBB$^+$13].

In academic contexts, almost always a self-contained problem definition is assumed: The processed scheduling horizon is regarded as isolated from previous and possible future time periods to be solved. Constraints on the number or total length of assignments are defined over the current period only (or parts thereof). If constraints dealing with sequences of assignments (e.g. maximum number of consecutive working days) are evaluated at the borders of the scheduling horizon at all, usually the surrounding time periods are assumed to be uniformly assigned (e.g. [HDCSS14]). In many cases (e.g. [BCP+08]), violations of sequence constraints at the beginning of the schedule are counted towards the objective function, while they are ignored at the end of the scheduling period (it is assumed that the schedule for the next period will be able to repair the sequences).

However, this usually does not correspond to real-world practices, where multiple interdependent schedules over consecutive time periods have to be generated and some constraints may span over more than one of these periods. In 2015, Ceschia et al. [CTC+15] introduced a new problem definition for the Second International Nurse Rostering Competition (INRC-II) to deal with this aspect. For this competition, solvers have to produce multiple schedules on a weekly basis, where the requirements of each week become known only after the schedule of the previous week has been fixed. In [SV12], such a setting was denoted a *stepping horizon* approach.

In such a multi-stage setting, it is necessary to consider the results of previous weeks when judging the quality of a schedule due to the dependencies between weeks. The details of this are already covered in the rules of the INRC-II and also treated in [GK10, SV12, SSV16].

An aspect that has not been explored by the previously mentioned publications, is that also future weeks have to be taken into account. Any solution approach that tries to find good solutions for each week without regard for the remaining stages will produce heavily imbalanced schedules that pay for minor improvements in the earlier stages with heavy penalties in the last weeks.

The aim of this thesis is to find methods suited for problems in a setting horizon setting, and in particular the one proposed for the INRC-II, that avoid these pitfalls and produce balanced schedules of a high quality.

Towards this end, first a new Integer Programming (IP) formulation for this problem is introduced according to the problem description in the rules of the competition. This basic model is then extended by additional constraints that model the dependencies between weeks and balance the conflicts over the whole period. While of course the developed constraints will be fitted to the specific requirements of the problem at hand, they should be general and modular enough to be applicable to a wide range of similar problems.

Further, to demonstrate the viability of this approach for various solution techniques, also a Local Search (LS) framework for this problem, based on a hybrid of Tabu Search and Min-Conflicts, is developed. This framework also makes use of the new constraints in its evaluation function to improve the quality of the generated schedules.

All solution approaches are evaluated on a set of benchmark instances provided for the INRC-II. It is shown that the solutions generated with models including the additional constraints have a significantly higher quality than those produced by treating each week as isolated from the rest. This holds both for the IP formulations and the LS framework, although the results for the former are noticeably better.

Finally, the results are also shown to be competitive to those of the 7 finalists in the INRC-II.

This thesis is structured as follows: Chapter 2 will provide an overview over publications dealing with staff scheduling in general and nurse rostering specifically. The theoretical background of the solution approaches used in this thesis will be described in Chapter 3. A formal definition of the problem will be given in Chapter 4, while an Integer Programming formulation for both the basic model and the developed extensions will be given in Chapter 5. Also the implementation of the Local Search framework will be described in this chapter. Chapter 7 contains experimental results with an evaluation of the impact of the various extensions and a comparison of the two solution approaches. Finally, a discussion of these results and possible future work can be found in Chapter 8.

CHAPTER 2

State of the Art

Staff scheduling in general and nurse rostering in particular have been active areas of research for multiple decades. As such, there is a wide range of publications dealing with problems from this field. This chapter gives an overview over selected publications and survey papers that highlight common themes and methods found in the literature.

2.1 Surveys

In the survey on staff scheduling by Ernst et al. [EJKS04] from 2004, the authors distinguish 6 different modules that together make up the complete staff scheduling process. These modules are demand modelling, days off scheduling, shift scheduling, line of work construction, task assignment and staff assignment. (Nurse rostering problems typically cover a combination of days off scheduling, line of work construction and staff assignment.) In the paper, problems are also grouped by application area and solution methods.

In 2013, Van den Bergh et al. [dBBB$^+$13] reviewed a list of 291 publications about staff scheduling and classified them according to problem specific qualities, solution methods and application area. While individual publications usually aren't described in detail, tables listing the categories and matching publications provide a fast and efficient way to find papers with certain properties or to compare the popularity of different features.

A survey from 2004 by Burke et al. [BCBL04] specifically deals with the nurse rostering problem. The authors compare a large number of publications according to various criteria, including both properties of the problem variants studied and solution methods employed in the publications.

De Causmaecker and Vanden Berghe [DCVB11] provided a categorization of different variants of the nurse rostering problems in 2011, analogous to the $\alpha|\beta|\gamma$ scheme for job scheduling problems. In their notation, the α part denotes properties of the problem

pertaining to individual (or groups of) nurses and constraints on their schedules, β describes the characteristics of the work and demand model and γ provides information about the optimization goals of the problem. A classification of various benchmark instances is given as an example. With their classification scheme, the authors hope to enable a systematic analysis of different problem features and their impact on various solution approaches and the complexity of the problem.

2.2 Solution approaches

Over the years, many different solution approaches from various fields of research have been applied to the Nurse Rostering Problem.

In the literature, there are two main categories of solution methods: Mathematical programming (usually integer programming) and heuristics [dBBB+13], although also other solution techniques have been applied.

2.2.1 Integer Programming

Most integer programming models use as their basis the Set Cover formulation developed by Dantzig [Dan54] in 1954, either implicitly or explicitly.

A fairly recent example is a work by Santos et al. [STGR14] from 2015, dealing with the problem posed in the first INRC [HDCSS14]. The authors develop and implement various cutting plane algorithms to improve dual bounds as well as a local search phase for fast generation and improvement of feasible solutions.

Especially when legal work patterns per nurse are listed explicitly, the number of necessary variables can make solving larger instances directly infeasible. In such cases, column generation is often applied, for example in [MS98] by Mason and Smith in 1998. Here, the authors first precalculate feasible work stretches and their cost for each nurse, which are later used in the column generation step to build roster lines.

Cappanera and Gallo [CG04] in 2004 used a different model to solve a variant of the related crew rostering problem for airlines. Their formulation is based on the multi-commodity flow problem, where each employee is represented by a commodity moving through a graph of possible assignments. Similar models are also often used for rerostering problems, where schedules need to be adapted to changing circumstances (e.g. [MP04]).

Also hybrid approaches combining integer programming with heuristic methods appear frequently. Burke et al. [BLQ10] use an IP model to find a feasible solution (using only the hard constraints) which they then optimize over the soft constraints using variable neighbourhood search.

2.2.2 Heuristics

Metaheuristic approaches are often based on various local search algorithms.

Of particular importance for nurse rostering is tabu search, which was used e.g. by Dowsland [Dow98] in 1998. This work is further of interest because the search repeatedly shifts its focus between satisfying the (hard) cover constraints and the (soft) constraints on shift patterns. This allows the exploration of larger parts of the search space by moving through otherwise infeasible solutions.

The work by Kundu et al. [KMMA08] from 2008 is exemplary for simulated annealing, the second metaheuristic commonly used for nurse rostering. In addition to that, the authors also implement and evaluate a genetic algorithm for the same problem.

In many recent works, a variable neighbourhood is employed to enable rapid improvements of a solution while also guiding the search away from local optima. An example of this strategy can be found in [BCP+08] from 2008 by Burke et al., where a second neighbourhood is explored as soon as the first cannot find any improving moves. The authors also implemented a construction heuristic to generate the initial schedule in which shifts are ranked according to the expected difficulty of assigning them and more problematic shifts are placed first.

Burke et al. [BCDCB01] in 2001 proposed a memetic algorithm, which performs an optimization step via local search before the recombination phase of a genetic algorithm. Such repair heuristics are often employed in combination with genetic algorithms, as the structure of staff rostering problems usually introduces extensive conflicts during the crossover of two solutions.

2.2.3 Other

In addition to those mentioned above, a large number of other techniques have been used to tackle nurse rostering problems, among them constraint satisfaction (e.g. [LLR03]), Boolean satisfiability testing ([KA08]), case-based reasoning ([PBVB03]) and neural networks ([HLT04]).

2.3 Multi-stage problems

Glass and Knight [GK10] first discussed the subject of continuity between scheduling periods in 2010. On the basis of the schedule for the previous month, they imposed specialized continuity constraints on the first days of the current month. Further, the authors used *implied penalties* - penalties that will necessarily appear in the solution of the next scheduling period due to the current schedule.

In 2012, Salassa and Vanden Berghe [SV12] introduced the concept of a *stepping horizon* to denote problem settings, where multiple scheduling periods have to be solved successively, with interdependencies between the periods and global constraints. Such a setting was later used for the INRC-II. Stepping horizon settings are contrasted with *static horizons*, where the problem is regarded as isolated from surrounding time periods, and *rolling horizons*, where the time window moves forward continuously, requiring repeated

rerostering of (parts of) the schedule. The authors explore a limited form of workload balancing between nurses in a stepping horizon setting (although the techniques used are just as applicable to static and rolling horizons) and show that the schedules of previous planning periods should be considered when generating a new schedule.

Smet et al. [SSV16] continued this work in 2016 by providing systematic integer programming formulations for various types of constraints which ensure that the objective function consistently takes previous planning periods into account. The techniques they propose are similar to those used in the rules of the INRC-II.

While not a multi-stage problem itself, Brunner et al. [BBK09] solved a complex scheduling problem by splitting it into multiple one-week periods which were solved separately. In each stage, the authors used information about overtime and weekend assignments of the previous week to generate a balanced and consistent schedule for the current week. Via this decomposition method, they could find optimal solutions even in cases where the monolithic model failed to generate feasible solutions at all.

A scientific treatment of the problem posed for the INRC-II can be found in [SFCO15] by Santos et al. from 2015. The problem is solved using a weighted constraint satisfaction approach. Despite several optimizations, feasible solutions could not be found for all instances and the penalty of the solutions is much higher than those of the finalists of the INRC-II and those achieved in this thesis. Also, no special consideration was placed on the stepping horizon setting, with each stage being solved independently. So far, this seems to be the only published work on this exact problem.

An abstract by Römer [RÏ5], the INRC-II competition winner, shows that they used an Integer Programming formulation based on multi-commodity network flows. However, the full publication is not yet available.

CHAPTER 3

Theoretical Background

The problem dealt with in this thesis is an *optimization problem*, where a solution (in this case a working schedule for a set of nurses) has to be found that satisfies a number of constraints while minimizing (or maximizing) an *objective function* F.

There are two main approaches to optimization problems: Heuristic algorithms and exact approaches. The latter guarantee that the produced solution is optimal with respect to the objective function, while the former give no such guarantee but typically generate good results faster than exact methods.

The rest of this chapter describes the two solution techniques used in this thesis: *Local search* is a heuristic optimization technique, while *(Integer) Linear programming* is an exact procedure.

3.1 Local Search

In Local Search, starting from an initial solution, small (*local*) changes are applied repeatedly until a termination criterion is reached. A single such change is called a *move* and the set of all solutions that can be generated from a given solution by applying a single move is the *neighbourhood* $\mathcal{N}$ of this solution.

The algorithm in its most general form (for a minimization problem) is described in Algorithm 3.1.

While the objective function is usually fixed for a given problem, the generation of the initial solution, the termination criterion and the strategy for selecting and applying moves can all be chosen freely. For the initial solution, simple random or greedy procedures are often used. The termination criterion usually is a time limit, a limit on the number of moves performed, a certain objective value that should be reached or any similar condition and combinations thereof. The most important part of the algorithm is the

Algorithm 3.1: Local Search

```
1  currentSolution ← GenerateInitialSolution();
2  bestSolution ← currentSolution;
3  while ¬TerminationCondition() do
4  |   m ← SelectMove(currentSolution);
5  |   currentSolution ← ApplyMove(currentSolution, m);
6  |   if F(currentSolution) < F(bestSolution) then
7  |   |   bestSolution ← currentSolution;
8  |   end
9  end
10 return bestSolution
```

selection of the change applied to the current solution at each iteration. This contains two main tasks: For once, the set of valid moves must be defined (and subsequently the neighbourhood of each solution). This usually depends on the problem structure. The other part is the selection of a move at each step to apply to the current solution. For this, a number of generally applicable strategies (*metaheuristics*) have been proposed.

One of the simplest is Hill-Climbing, where at each step, the move that results in the best objective value is calculated and applied. However, this strategy tends to get stuck in local optima.

One way to avoid this situation is to restart the search from a new initial solution as soon as the current solution can no longer be improved.

Many other, more refined techniques exist and are widely used for many different problems. The two strategies used in this thesis are *Tabu Search* and *Min-Conflict*.

3.1.1 Tabu Search

Tabu Search was first introduced by Glover [Glo89]. Naive Hill-Climbing will either stop in a local optimum or, if non-improving moves are allowed, oscillate around this optimum, repeating the same solutions and moves multiple times. To force the algorithm to explore new regions of the search space, re-visiting recently seen solutions is discouraged. This is done by collecting previously applied moves in a *tabu list* and allowing only moves that do not already appear on this list (i.e. are *tabu*). After a certain time, the oldest moves in the tabu list are removed from the list to avoid the algorithm getting stuck in an unfavourable region of the search space.

A description of Tabu Search can be found in Algorithm 3.2.

Note that at each iteration, it is possible that the move chosen does not improve the current solution. This is necessary to escape from local optima.

The length of the tabu list depends on the size and structure of the problem. If the list is

Algorithm 3.2: Tabu Search

```
1  currentSolution ← GenerateInitialSolution();
2  bestSolution ← currentSolution;
3  tabuList ← {};
4  while ¬TerminationCondition() do
5  |  m ← best move that is not tabu;
6  |  tabuList.add(m);
7  |  if tabuList.size() > k then
8  |  |  tabuList.removeOldest();
9  |  end
10 |  currentSolution ← ApplyMove(currentSolution, m);
11 |  if F(currentSolution) < F(bestSolution) then
12 |  |  bestSolution ← currentSolution;
13 |  end
14 end
15 return bestSolution
```

too short, the algorithm can get stuck in cycles. If it is too long, paths to good solutions might be blocked by older moves that are still in the tabu list.

A further refinement is adding an *aspiration criterion* that allows moves to be accepted even if they are tabu. A common such criterion is accepting a move if the generated solution is better than the previously best solution.

Other commonly used variants include adding both the accepted move and the inverse move to the tabu list, to prevent the algorithm from reversing a change immediately, or varying the length of the tabu list depending on the progress of the search.

3.1.2 Min-Conflict

Another commonly used way to escape local optima is the use of randomness. One heuristic that does so is the Min-Conflict heuristic, introduced by Minton et al. [MJPL92].

At each iteration, a variable in the solution is chosen at random. Then only those moves that involve the chosen variable are evaluated and the one that minimizes the remaining conflicts is applied. As before for Tabu Search, also non-improving moves are allowed to be able to escape local optima.

In order to avoid disrupting parts of the solution that already fulfil all constraints, the variable that is chosen must appear in some conflict.

Due to the random choice of a variable and the restricted subset of the neighbourhood that is regarded at each step, cycles are avoided and the solution eventually moves away from local optima.

Algorithm 3.3: Min-Conflict

```
currentSolution ← GenerateInitialSolution();
bestSolution ← currentSolution;
while ¬TerminationCondition() do
    v ← random variable that appears in a conflict;
    m ← best move involving v;
    currentSolution ← ApplyMove(currentSolution, m);
    if F(currentSolution) < F(bestSolution) then
        bestSolution ← currentSolution;
    end
end
return bestSolution
```

A disadvantage is that in situations where a particular move would lead to a substantial improvement, this might not be detected due to the randomness involved. However, the reduced number of moves that have to be evaluated for each step means that for a given time limit, a much greater number of moves can be performed than for Tabu Search.

3.2 (Integer) Linear Programming

The solution space of optimization problems typically is a set S of tuples corresponding to feasible variable assignments, with the domain of each variable a subset of $\mathbb{R}$. Therefore, the solution space for a problem with n variables can be interpreted as a subset of $\mathbb{R}^n$.

The set S is usually described via constraints between variables that all elements have to satisfy (including the domain restrictions). Depending on the properties of these constraints (and the objective function), various different solution strategies exist, both exact and heuristic.

An exact method that can be used if the objective function is a linear combination of the variables and all constraints are linear (in)equalities is *Linear Programming*.

3.2.1 Linear Programming

A linear program (LP) has the following canonical form:

$$\begin{aligned} &\text{maximize} && \\ &&& \mathbf{c}^T\mathbf{x} \\ &\text{subject to} && \\ &&& \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ &&& \mathbf{x} \geq \mathbf{0} \end{aligned}$$

where $\mathbf{x}$ are the unknown variables, $\mathbf{c}$ and $\mathbf{b}$ are vectors and $\mathbf{A}$ is a coefficient matrix.

In a more general form, also inequalities with $\geq$, equalities and (potentially) negative variables are allowed as well as minimization of the objective function. However, each LP in a general form can be transformed into an equivalent one in canonical form.

Example 1. *An example of an LP with two variables is the following:*

$$
\begin{aligned}
&\textit{maximize} \\
&\quad 2x + y \\
&\textit{subject to} \\
&-x + y \leq 2 \\
&2x + 3y \leq 14 \\
&5x + y \leq 20 \\
&x, y \geq 0
\end{aligned}
$$

All values for x and y that fulfil all three constraints are feasible solutions, while $x = 46/13$ and $y = 30/13$ is the optimal solution that maximizes the objective value.

A graphical representation of this LP can be seen in Figure 3.1.

In general, the feasible region of a LP with n variables is an n-dimensional convex polyhedron.

While research in the area of optimization in the presence of linear constraints existed already in the 19th century, the term *Linear Programming* was coined by Dantzig in 1946 [Dan02]. Dantzig also developed the *Simplex algorithm*, which is still the most widely used algorithm for solving linear programs.

The Simplex algorithm is based on the fact that in a linear program, any local optimum is also a global optimum. Therefore, starting from any feasible solution at a vertex of the polyhedron, it is always possible to move along an edge in the direction of increasing objective value (for a maximization problem), unless the optimum is already reached.

There are a few problems with the base form of the Simplex algorithm (which have by now been solved by extensions to the original version), most notably the potential for stalling or cycling in the case where multiple boundaries join at the same vertex. Also the generation of the initial solution is not trivial in most cases.

The Simplex algorithm has an exponential worst-case complexity in degenerate cases, but in practice it is able to find optimal solutions very fast.

There are also algorithms that can solve LPs in polynomial time (including the *Ellipsoid Method* and *Interior Point Methods*), but they are mostly not competitive to efficient implementations of the Simplex algorithms.

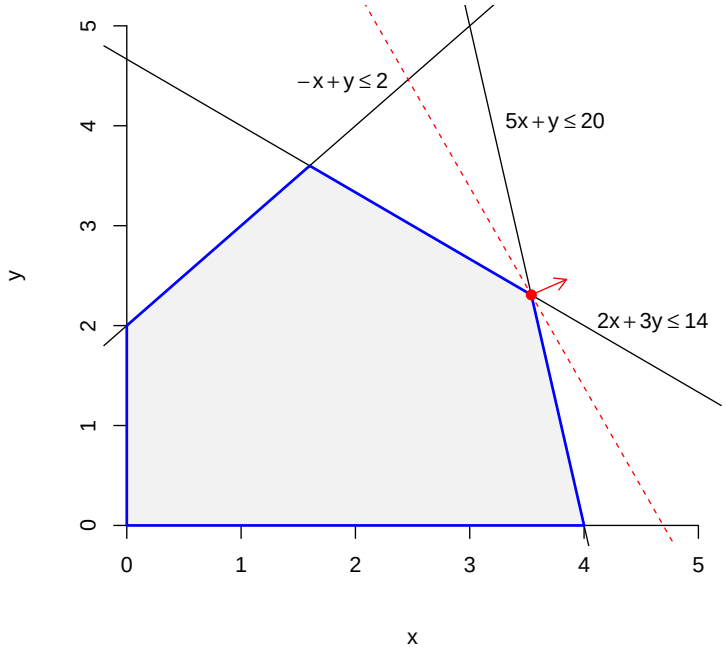

Figure 3.1: Graphical representation of the linear program in Example 1. Each point inside the area outlined in blue (edges inclusive) is a feasible solution to the LP. All points on the dashed red line have the same objective value, with the value increasing in the direction of the arrow. It is easy to see that the marked point is the optimum solution.

A further improvement from duality theory was already proposed by Von Neumann in 1947 [Dan02]. Each linear program (the *primary*) can be directly transformed into a *dual* linear program with the following qualities: The dual of a maximization problem is a minimization problem (and vice versa) and the optimum solution to the dual is equal to the optimum solution for the original problem. Therefore, finding any feasible solution for the dual program immediately gives an upper bound (for a maximization problem) for the optimum solution of the primary program.

3.2.2 Integer Programming

While LPs can theoretically be solved in polynomial time, this is no longer the case if the variables can only take on integer values. *Integer Linear Programs* (IPs) have the following canonical form:

$$
\begin{aligned}
&\text{maximize} && \\
&\qquad && \mathbf{c}^T\mathbf{x} \\
&\text{subject to} && \\
&\qquad && \mathbf{A}\mathbf{x} \leq \mathbf{b} \\
&\qquad && \mathbf{x} \geq \mathbf{0} \\
&\qquad && \mathbf{x} \in \mathbb{Z}^n
\end{aligned}
$$

As before, IPs in general form also allow minimization, $\geq$ constraints and unbounded variables and can be translated to an equivalent IP in canonical form.

Given an IP, the LP that arises from removing the integrality constraints is called the *LP-relaxation*. It is easy to see that all solutions of the original IP are also solutions for the LP-relaxation. However, the converse is not true and in particular the optimum solution is different in general.

Example 2. *If Example 1 is extended to require x and y to be integer, the corresponding IP looks like this:*

$$
\begin{aligned}
&\textit{maximize} && \\
&\qquad && 2x + y \\
&\textit{subject to} && \\
&\qquad -x + y && \leq 2 \\
&\qquad 2x + 3y && \leq 14 \\
&\qquad 5x + y && \leq 20 \\
&\qquad x, y && \geq 0 \\
&\qquad x, y && \in \mathbb{Z}
\end{aligned}
$$

The graphical representation of this IP is shown in Figure 3.2.

In contrast to the previous example, the integer problem has two optimum solutions - $(3, 2)$ and $(4, 0)$, which are both different from the optimum solution of the LP-relaxation.

The problem of finding, whether any feasible solution for a given IP exists, is already NP-hard [PS82] and subsequently also the problem of finding the best solution. In practice, the solution of the LP-relaxation is usually used as a starting point and upper bound (for maximization problems) for a Branch-and-Bound search over the fractional variables. Competitive solvers also make use of the dual problem, additional inequalities that reduce the search space without cutting off integer solutions and various heuristics to find feasible solutions and improve the bounds on the optimum solution.

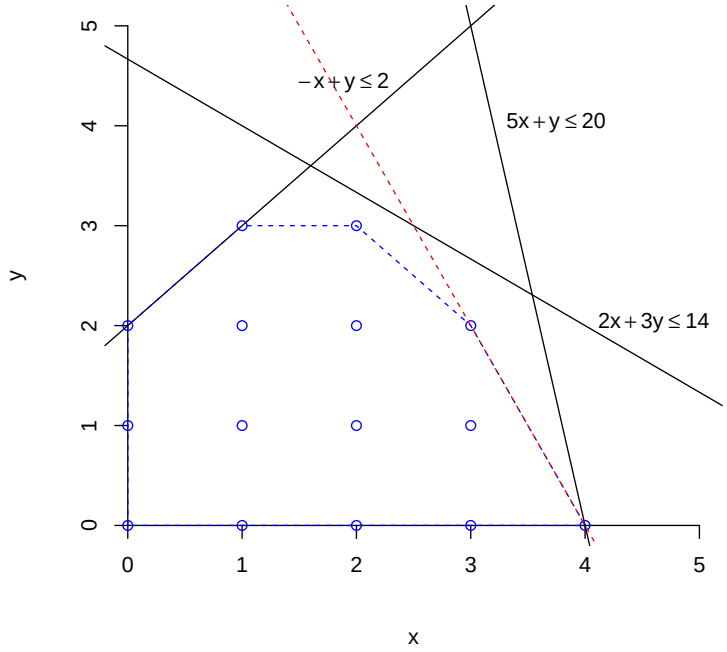

Figure 3.2: Graphical representation of the IP in Example 2. The feasible solutions are all points at integer coordinates within the feasible region of the LP relaxation.

3.2.3 Staff scheduling formulations

While there are many variations of staff scheduling problems, nearly all of them have at their core the requirement to assign employees to various tasks over the course of a certain scheduling period. As a consequence, many IP formulations for these problems are founded on similar base models that are adapted to the problem variation at hand[dBBB+13].

The most widely used type of model is based on a formulation first used by Dantzig in 1954 [Dan54]. This formulation uses the Set Cover problem to solve assignments of toll booth operators as described in [Edi54].

Given a set of $n + T$ timeslots, and a demand for b_t operators at timeslot t, we find the minimum number of operators necessary using the following LP:

$$
\begin{aligned}
&\text{minimize} \\
&\quad \sum_{j=0}^{n} x_j \\
&\text{subject to} \\
&\sum_{j=0}^{n} a_{tj} x_j \;\geq\; b_t && 1 \leq t \leq n+T \\
&\qquad x_j \;\geq\; 0 && 0 \leq j \leq n
\end{aligned}
$$

where the decision variable x_j denotes the number of operators starting their work at time j and a_{tj} is 1 if an operator starting at time j is working during timeslot t and 0 if they are on break or the timeslot lies outside their working hours.

This model, also known as the explicit set covering formulation[CGQR11], becomes unwieldy if more flexible working patterns are introduced (as each possible pattern must be represented by a new decision variable). For this reason, it is mostly used in column generation approaches, where only a subset of patterns is regarded at any time and new decision variables are added dynamically.

In contrast, the implicit set covering formulation handles different characteristics of work patterns (such as start/end times and break assignments) separately. This reduces the number of decision variables, but also requires additional constraints to ensure that the generated solutions are feasible.

A variation proposed already by Dantzig involves splitting the rigid work patterns into the individual timeslots and using the decision variables to denote whether an operator works during a particular slot. This is closer to the formulation used for many modern rostering problems and also in this thesis. Modern formulations usually split work patterns into timeslots of either fixed length (including whole days), shifts or individual tasks to be performed. This is known as the compact assignment formulation, because it usually needs a smaller number of variables compared to the above formulation.

In cases where work patterns need to be assigned to individual employees (as in most modern nurse rostering problems), two- (or higher) dimensional decision variables denote the correspondence between employees and assigned work patterns, in either of the two forms described above.

A different, rather modern, class of IP models use formulations based on network flow problems. For example, Balakrishnan et al. [BW90] used a model where the days of the scheduling period are represented as nodes of the network and arcs between them correspond to feasible work and rest periods. The various constraints of the problem are incorporated into the structure of the model such that a complete solution is represented by a path through the network from the source (the beginning of the first day) to the sink (the end of the last day).

While very good results have been achieved using such network based formulations (including the winners of the INRC-II [RÏ5]), many constraints are hard to model (in the case of [BW90], coverage constraints had to be modelled separately from the network formulation) and it is often difficult to adapt these models to new problem variants.

CHAPTER 4

Problem Definition

The task given for the INRC-II problem (defined in [CTC+15]) is to produce a series of either 4 or 8 sequential weekly schedules (stages). The parameters for each schedule are stored in three separate files:

- A *scenario* file, containing general parameters about the shifts, skills, nurses and contracts available.
- *Week* data with the cover requirements for the current stage and individual nurse preferences.
- A *history* file including relevant information about assignments at the end of the previous week.

While the global scenario stays the same over all stages of an instance, the other files are different between stages. In particular, the week data is known only for the current stage (although solvers are not prohibited from storing additional information about previous weeks). The history file is given as part of the instance for the first stage and generated from the solution of the previous week for all subsequent stages.

Solutions should satisfy four hard constraints and seven soft constraints at varying weights. Two of these are global constraints that are evaluated over all stages at the end. The total penalty summed up over all stages should be minimized.

4.1 Constraints

In the following, a *work stretch* denotes a period of consecutive working days for a nurse. *Rest stretch* and *shift stretch* are analogously defined for periods of consecutive days off and assignments to the same shift, respectively.

All solutions have to satisfy the following hard constraints:

H1. Single assignment per day Each nurse can only work a single shift using a single skill per day.

H2. Under-staffing The minimum number of nurses required for each shift and skill must be present on each day.

H3. Shift type successions Nurses must not have shifts on two consecutive days that form a forbidden sequence.

H4. Missing required skill Nurses can only cover assignments for which they have the required skill.

Forbidden sequences (H3) are defined on a per-scenario basis, but usually include earlier shifts following later shifts (such as a Day shift after a Night shift).

In addition to these constraints, there are also soft constraints, whose violation incurs a penalty to the objective function. The weight of each constraint is given in brackets after the name, this is multiplied by the degree of violation for most constraints.

S1. Insufficient staffing for optimal coverage (30) The number of nurses assigned to each shift and skill should not be smaller than the optimum staffing.

S2. Consecutive assignments (15/30) The length of each shift stretch (weight 15) and work stretch (weight 30) should be within the bounds defined for the shift type resp. the contract of the involved nurse.

S3. Consecutive days off (30) As before, the length of each rest stretch should be within the bounds defined in each nurse's contract.

S4. Preferences (10) The requests of nurses for shifts (or days) off should be respected.

S5. Complete week-end (30) Nurses with the complete-weekend constraint in their contract should either work both days of the weekend or none.

S6. Total assignments (20) Over the whole planning horizon, each nurse's assignments should be within the bounds defined in their contract.

S7. Total working week-ends (30) Over the whole planning horizon, each nurse should not work more than the maximum number of weekends defined in their contract.

Constraints H3, S2 and S3 are *sequence constraints*. Their evaluation involves the history data from the previous week. In general, sequences starting in the previous week are evaluated as normal, while penalties at the end of the week are counted only for already

	Fr	Sa	Su	Mo	
...	N	N	N	N	...
	...	-	N	-	...

(a)

	We	Th	Fr	Sa	Su	Mo	
...	-	N	N	N	N	?	...
			...	-	N	?	...

(b)

Figure 4.1: Example of the evaluation of sequence constraints at the beginning (a) and end (b) of the week, assuming a maximum length of night stretches (N) of 3 and a minimum of 2. Red and blue cells mark violations of maximum and minimum shift stretch length, respectively. Note that the second sequence in (b) does not incur a penalty, because it could be extended to a valid length in the next stage.

known violations of the maximum length part of S2 and S3. Examples of the evaluation can be seen in Figure 4.1, a more detailed description is given in [CTC+15].

The last two constraints, S6 and S7, are the two global constraints. Counters for the already assigned shifts and working weekends are passed from stage to stage via the history file. After all stages have been solved, the values of these counters are compared to the limits specified in the contract of each nurse and, if they are outside the bounds, the appropriate penalty is added to the objective value.

4.2 Parameters

As mentioned before, parameters for each stage are passed via three separate files.

The first set of parameters, stored in the scenario file, deals with general information about shifts, skills and the nurses and their contracts:

N	set of nurses
S	set of shifts
K	set of skills
$\|W\|$	number of weeks
$w_n^{[+/-]}$	maximum/minimum consecutive working days for nurse n
$f_n^{[+/-]}$	maximum/minimum consecutive days off for nurse n
$a_n^{[+/-]}$	maximum/minimum total assignments for nurse n across planning horizon
t_n^+	maximum number of working weekends for nurse n across planning horizon
b_n	boolean, 1 iff either both days of a weekend should be worked by nurse n, or none
κ_{nk}	boolean, 1 iff nurse n has skill k
$\sigma_s^{[+/-]}$	maximum/minimum consecutive assignments of shift s
u_{st}	boolean, 1 iff shift t may be assigned the day after an assignment of shift s

These values stay the same for all stages of an instance.

Further, the coverage requirements as well as the weekly requests of the nurses are stored in the week data, which can be different between stages:

w	number of the current week
c_{sk}^d	minimum cover requirements for day d, shift s and skill k
o_{sk}^d	optimum cover requirements for day d, shift s and skill k
r_{ns}^d	boolean, 1 iff nurse n requested not to work in shift s on day d ($s = 0$ is day-off request)

Finally, the last set of parameters is from the history and contains global counters as well as information about the last assignments of each nurse in the previous week:

l_n^{id}	id of last shift worked by nurse n in previous week (0 if day off)
l_{ns}	consecutive shifts of type s worked by nurse n at the end of the previous week (0 if $s \neq l_n^{id}$)
l_n^w	consecutive working days for nurse n at the end of the previous week (0 if $l_n^{id} = 0$)
l_n^f	consecutive days off for nurse n at the end of the previous week (0 if $l_n^{id} \neq 0$)
a_n^{tot}	total number of assignments for nurse n so far
t_n^{tot}	total number of weekends worked by nurse n so far

CHAPTER 5

Integer Programming Formulations

This chapter describes the formulation of the INRC-II problem as an IP, on the basis of the compact assignment formulation discussed in Section 3.2.3.

First, an implementation of the problem as described in Chapter 4 is provided as a starting point. This model yields an optimal solution when isolated single stages are regarded. Based on this basic model, extensions to deal with the stepping horizon setting are introduced and discussed. These extensions take the form of additional (soft) constraints that can be added to the model independently.

5.1 Basic Model

5.1.1 Decision variables

There are three types of decision variables. The basis of the model is formed by a set of boolean variables indicating the schedule itself:

$$\forall n \in N, s \in S, k \in K, d \in \{1 \dots 7\}$$
$$x^d_{nsk} = \begin{cases} 1 & \text{if nurse } n \text{ is assigned to shift } s \text{ using skill } k \text{ on day } d \\ 0 & \text{otherwise} \end{cases}$$

In order to deal with constraints S5 (Complete weekends) and S7 (maximum number of

working weekends), an additional set of variables is necessary:

$$\forall n \in N$$
$$W_n = \begin{cases} 1 & \text{if nurse } n \text{ works on at least one day of the weekend} \\ 0 & \text{otherwise} \end{cases}$$

The last set of decision variables consists of surplus variables for the soft constraints. Whenever one of these constraints would be violated, the corresponding surplus variable can be set to a value > 0 to deactivate the constraint. However, the surplus variables are included in the objective function and therefore each use incurs a penalty.

C^{S1}_{skd}	≥ 0	missing nurses for optimal coverage of shift s, skill k on day d
C^{S2a}_{nsd}	≥ 0	missing days in the block of shifts s starting on day d for nurse n
C^{S2b}_{nsd}	$\in \{0,1\}$	1 iff shift s of nurse n on day d violates maximum consecutive shifts
C^{S2c}_{nd}	≥ 0	missing days in the work block of nurse n starting on day d
C^{S2d}_{nd}	$\in \{0,1\}$	1 iff work of nurse n on day d violates maximum consecutive work days
C^{S3a}_{nd}	≥ 0	missing days in the rest stretch of nurse n starting on day d
C^{S3b}_{nd}	$\in \{0,1\}$	1 iff day off of nurse n on day d violates maximum consecutive days off
C^{S4}_{nd}	$\in \{0,1\}$	1 iff assignment on day d violates a request of nurse n
C^{S5}_{n}	$\in \{0,1\}$	1 iff nurse n violates complete weekend constraint
C^{S6}_{n}	≥ 0	number of total shifts outside the allowed bounds for nurse n
C^{S7}_{n}	≥ 0	number of weekends worked above the maximum by nurse n

Depending on the nature of their associated constraints, some of the surplus variables are binary, while others can take any non-negative value. This is particularly of note in the case of the sequence constraints (S2, S3), where surplus variables have different domains depending on whether they deal with violations of the maximum or minimum sequence length.

5.1.2 Objective function

The objective function is a weighted sum of all surplus variables that were necessary to deactivate otherwise violated (soft) constraints.

$$
\begin{aligned}
minimize\ f = \quad & 30 * \sum_{\substack{s\in S\\ k\in K\\ d\in\{1...7\}}} C^{S1}_{skd} \\
& +15 * \sum_{\substack{n\in N\\ s\in S\\ d\in\{1...7\}}} (C^{S2a}_{nsd} + C^{S2b}_{nsd}) \\
& +30 * \sum_{\substack{n\in N\\ d\in\{1...7\}}} (C^{S2c}_{nd} + C^{S2d}_{nd}) \\
& +30 * \sum_{\substack{n\in N\\ d\in\{1...7\}}} (C^{S3a}_{nd} + C^{S3b}_{nd}) \\
& +10 * \sum_{\substack{n\in N\\ d\in\{1...7\}}} C^{S4}_{nd} \\
& +30 * \sum_{n\in N} C^{S5}_{n} \\
& +20 * \sum_{n\in N} C^{S6}_{n} \\
& +30 * \sum_{n\in N} C^{S7}_{n}
\end{aligned}
$$

5.1.3 Constraints

The following (in)equalities model the hard constraints, as described above.

$$
\begin{aligned}
&\text{H1} \quad \forall n \in N, d \in \{1 \dots 7\} \\
&\qquad \sum_{\substack{s\in S\\ k\in K}} x^d_{nsk} \leq 1 \qquad (5.1) \\
&\text{H2} \quad \forall s \in S, k \in K, d \in \{1 \dots 7\} \\
&\qquad \sum_{n\in N} x^d_{nsk} \geq c^d_{sk} \qquad (5.2)
\end{aligned}
$$

For constraint H3, any forbidden shift sequence ($u_{s_1 s_2} = 0$) must not be assigned to the same nurse on consecutive days. This must be ensured both within the week (a) and at the boundary of this week with the previous one (i.e. on the first day of the week, b).

$$
\begin{aligned}
&\text{H3a} \quad \forall n \in N, s_1, s_2 \in S, k \in K, d \in \{1 \dots 6\} : u_{s_1 s_2} = 0 \\
&\qquad \sum_{k\in K} x^d_{ns_1k} + \sum_{k\in K} x^{d+1}_{ns_2k} \leq 1 \qquad (5.3) \\
&\text{H3b} \quad \forall n \in N, s \in S, k \in K : u_{l^{id}_n s} = 0 \\
&\qquad x^1_{nsk} = 0 \qquad (5.4)
\end{aligned}
$$

$$\begin{aligned} \text{H4} \quad & \forall n \in N, s \in S, d \in \{1 \dots 7\}, k \in K : \kappa_{nk} = 0 \\ & x^d_{nsk} = 0 \end{aligned} \tag{5.5}$$

The remaining inequalities deal with the soft constraints. The right hand side of each constraint contains the corresponding surplus variable which can be used to deactivate the constraint (incurring a penalty in the objective function).

$$\begin{aligned} \text{S1} \quad & \forall s \in S, k \in K, d \in \{1 \dots 7\} \\ & \sum_{n \in N} x^d_{nsk} \geq o^d_{sk} - C^{S1}_{skd} \end{aligned} \tag{5.6}$$

Since none of the other soft constraints deal with skills, all following inequalities have their left hand side summed up over all skills.

S2 actually contains various different constraints that have to be modeled separately: consecutive assignments of the same shift (min (a)/ max (b)) and of work in general (min (c) / max (d)), both during and at the start of the week.

For the minimum consecutive shifts constraints, all patterns that compose a sequence shorter than the required length are prevented. For example, if the minimum number of consecutive night shifts (N) is 4, the patterns {xNx, xNNx, xNNNx}, where x is any other shift or a day off, should not appear.

Since each pattern incurs a penalty proportional to the number of missing assignments, (in the example, xNx would incur a penalty of 45, while xNNNx would incur a penalty of 15) the surplus variables are weighted correspondingly, to ensure that a value of at least the number of missing assignments is necessary to deactivate the constraint.

Equations 5.8 model the case where a stretch starts at the beginning of the week or towards the end of the previous week. For the latter case, this stretch should not end immediately if it does not yet have the minimum length. This is handled by Equations 5.9.

$$\begin{aligned} \text{S2a} \quad & \forall s \in S, n \in N, b \in \{1 \dots (\sigma^-_s - 1)\}, d \in \{1 \dots 7 - (b+1)\} \\ & \sum_{k \in K} (x^d_{nsk} + \sum_{i \in \{1 \dots b\}} (1 - x^{d+i}_{nsk}) + x^{d+b+1}_{nsk}) \geq 1 \; - \; \frac{C^{S2a}_{ns(d+1)}}{\sigma^-_s - b} \end{aligned} \tag{5.7}$$

$$\begin{aligned} & \forall s \in S, n \in N, b \in \{1 \dots (\sigma^-_s - 1 - l_{ns})\} \\ & \sum_{k \in K} (\sum_{i \in \{1 \dots b\}} (1 - x^i_{nsk}) + x^{b+1}_{nsk}) \geq 1 \; - \; \frac{C^{S2a}_{ns1}}{\sigma^-_s - l_{ns} - b} \end{aligned} \tag{5.8}$$

$$\begin{aligned} & \forall s \in S, n \in N : l^{id}_n = s \wedge l_{ns} < \sigma^-_s \\ & \sum_{k \in K} x^1_{nsk} \geq 1 \; - \; \frac{C^{S2a}_{ns1}}{\sigma^-_s - l_{ns}} \end{aligned} \tag{5.9}$$

The maximum consecutive shifts constraints is modeled like this: For each shift s with a maximum of σ_s^+ consecutive assignments, each block of $\sigma_s^+ + 1$ days must contain at least one day where s is not assigned. Note that contrary to the situation for S2a, violations of this constraint by more than one shift assignment result in multiple matches of the pattern and therefore it suffices to use boolean surplus variables.

As before, equations 5.11 model the case where a shift block started in the previous week.

S2b $\forall s \in S, n \in N, d \in \{1 \dots (7 - \sigma_s^+)\}$

$$\sum_{k \in K} \sum_{i \in \{0 \dots \sigma_s^+\}} x_{nsk}^{d+i} \leq \sigma_s^+ \; + \; C_{ns(d+\sigma_s^+)}^{S2b} \tag{5.10}$$

$\forall s \in S, n \in N, b \in \{(\sigma_s^+ - l_{ns} + 1) \dots \sigma_s^+\} : l_n^{id} = s$

$$\sum_{k \in K} \sum_{i \in \{1 \dots b\}} x_{nsk}^{i} \leq b - 1 \; + \; C_{nsb}^{S2b} \tag{5.11}$$

The inequalities modelling the maximum and minimum length of work stretches (S2c, S2d) function analogously to those for shift stretches. The only difference is that an assignment to any shift counts towards the length of the work stretch.

S2c $\forall n \in N, b \in \{1 \dots (w_n^- - 1)\}, d \in \{1 \dots 7 - (b+1)\}$

$$\sum_{\substack{s \in S \\ k \in K}} (x_{nsk}^{d} + \sum_{i \in \{1 \dots b\}} (1 - x_{nsk}^{d+i}) + x_{nsk}^{d+b+1}) \geq 1 \; - \; \frac{C_{n(d+1)}^{S2c}}{w_n^- - b} \tag{5.12}$$

$\forall n \in N, b \in \{1 \dots (w_n^- - 1 - l_n^w)\}$

$$\sum_{\substack{s \in S \\ k \in K}} (\sum_{i \in \{1 \dots b\}} (1 - x_{nsk}^{i}) + x_{nsk}^{b+1}) \geq 1 \; - \; \frac{C_{n1}^{S2c}}{w_n^- - l_n^w - b} \tag{5.13}$$

$\forall n \in N : l_n^{id} \neq 0 \wedge l_n^w < w_n^-$

$$\sum_{\substack{s \in S \\ k \in K}} x_{nsk}^{1} \geq 1 \; - \; \frac{C_{n1}^{S2c}}{w_n^- - l_n^w} \tag{5.14}$$

S2d $\forall n \in N, d \in \{1 \dots (7 - w_n^+)\}$

$$\sum_{\substack{s \in S \\ k \in K}} \sum_{i \in \{0 \dots w_n^+\}} x_{nsk}^{d+i} \leq w_n^+ \; + \; C_{n(d+w_n^+)}^{S2d} \tag{5.15}$$

$\forall n \in N, b \in \{(w_n^+ - l_n^w + 1) \dots w_n^+\} : l_n^{id} \neq 0$

$$\sum_{\substack{s \in S \\ k \in K}} \sum_{i \in \{1 \dots b\}} x_{nsk}^{i} \leq b - 1 \; + \; C_{nb}^{S2d} \tag{5.16}$$

S3 similarily contains two independent constraints: the minimum (a) and maximum (b) number of consecutive days off, again both during and at the start of the week.

The equations modelling these constraints are again analoguous to those from constraints S2c and S2d, except that days of work and days off were swapped.

S3a $\forall n \in N, b \in \{1 \dots (f_n^- - 1)\}, d \in \{1 \dots 7 - (b+1)\}$

$$\sum_{\substack{s \in S \\ k \in K}} ((1 - x_{nsk}^d) + \sum_{i \in \{1 \dots b\}} x_{nsk}^{d+i} + (1 - x_{nsk}^{d+b+1})) \geq 1 \; - \; \frac{C_{n(d+1)}^{S3a}}{f_n^- - b} \tag{5.17}$$

$\forall n \in N, b \in \{1 \dots (f_n^- - 1 - l_n^f)\}$

$$\sum_{\substack{s \in S \\ k \in K}} (\sum_{i \in \{1 \dots b\}} x_{nsk}^i - x_{nsk}^{b+1}) \geq 0 \; - \; \frac{C_{n1}^{S3a}}{f_n^- - l_n^f - b} \tag{5.18}$$

$\forall n \in N : l_n^{id} = 0$

$$\sum_{\substack{s \in S \\ k \in K}} -x_{nsk}^1 \geq 0 \; - \; \frac{C_{n1}^{S3a}}{f_n^- - l_n^f} \tag{5.19}$$

S3b $\forall n \in N, d \in \{1 \dots (7 - f_n^+)\}$

$$\sum_{\substack{s \in S \\ k \in K}} \sum_{i \in \{0 \dots f_n^+\}} x_{nsk}^{d+i} \geq 1 \; - \; C_{n(d+f_n^+)}^{S3b} \tag{5.20}$$

$\forall n \in N, b \in \{(f_n^+ - l_n^f + 1) \dots f_n^+\} : l_n^{id} = 0$

$$\sum_{\substack{s \in S \\ k \in K}} \sum_{i \in \{1 \dots b\}} x_{nsk}^i \geq 1 \; - \; C_{nb}^{S3b} \tag{5.21}$$

To model nurse requests for shifts or days off, any assignment to an unwanted shift incurs the penalty.

S4 $\forall n \in N, s \in S, d \in \{1 \dots 7\} : r_{ns}^d \vee r_{n0}^d$

$$\sum_{k \in K} x_{nsk}^d \leq C_{nd}^{S4} \tag{5.22}$$

For the complete weekends constraint, first the additional helper variables W_n are set if the nurse n works either of the days on the weekend. Equations 5.24 then ensure that if W_n is set, and the complete weekend constraint is present for the nurse, both days of the weekend should have work assigned.

S5 $\forall n \in N, d \in \{6, 7\}$

$$\sum_{\substack{s \in S \\ k \in K}} x_{nsk}^d \leq W_n \tag{5.23}$$

$\forall n \in N : b_n = 1$

$$\sum_{\substack{s \in S \\ k \in K}} (x_{nsk}^6 + x_{nsk}^7) \geq 2W_n \; - \; C_n^{S5} \tag{5.24}$$

The constraint S6 (number of total assignments) is modeled slightly differently from its description in [CTC+15]. Originally, these constraints were evaluated only after the schedules of all weeks were fixed. In this model, the penalties are calculated immediately and added to the objective function value of the week in which they arise. This does not change the overall quality of the whole schedule, so results are still comparable, although the intermediate quality value of the individual weeks might be different.

$$\text{S6} \quad \forall n \in N \quad \sum_{\substack{s \in S \\ k \in K \\ d \in \{1...7\}}} x^d_{nsk} \leq \max\{a^+_n - a^{tot}_n,\ 0\} \ +\ C^{S6}_n \tag{5.25}$$

$$\forall n \in N \quad \sum_{\substack{s \in S \\ k \in K \\ d \in \{1...7\}}} x^d_{nsk} \geq \min\{a^-_n - 7 * (|W| - w),\ 7\} \ -\ C^{S6}_n \tag{5.26}$$

The equations for constraint S7 (maximum number of weekends worked) use the variable W_n, set in equations 5.23.

$$\text{S7} \quad \forall n \in N \quad t^{tot}_n + W_n \leq t^+_n \ +\ C^{S7}_n \tag{5.27}$$

5.2 Model Extensions

The basic model produces solutions that are optimal if each stage is regarded in isolation. However, when multiple stages have to be solved in succession, it turned out that the solution quality of each individual stage is heavily imbalanced, favouring earlier weeks. This effect is demonstrated in Figure 5.1.

One reason for this is that constraints S6 (total assignments) become relevant only during the later weeks. An additional factor is that the solutions of earlier weeks are generated without regard to the requirements of later stages. As a result, good solutions in the first weeks are bought with disproportionally higher penalties in the last weeks, especially regarding the global constraints and sequence constraints at the boundaries between weeks.

To mitigate these problems, the basic model has to be extended with additional considerations that also take future stages into account.

These additional, “artificial” constraints will be included in the optimization step of the solver. Once a solution has been fixed, the actual penalty has to be recalculated according to the basic model to get the correct results. During the last week, the artificial constraints should be ignored altogether, as there are no more future weeks to take care of. Of course, this is only permissible in an academic setting like this, whereas in real world applications, the solution for all weeks should use the extended model.

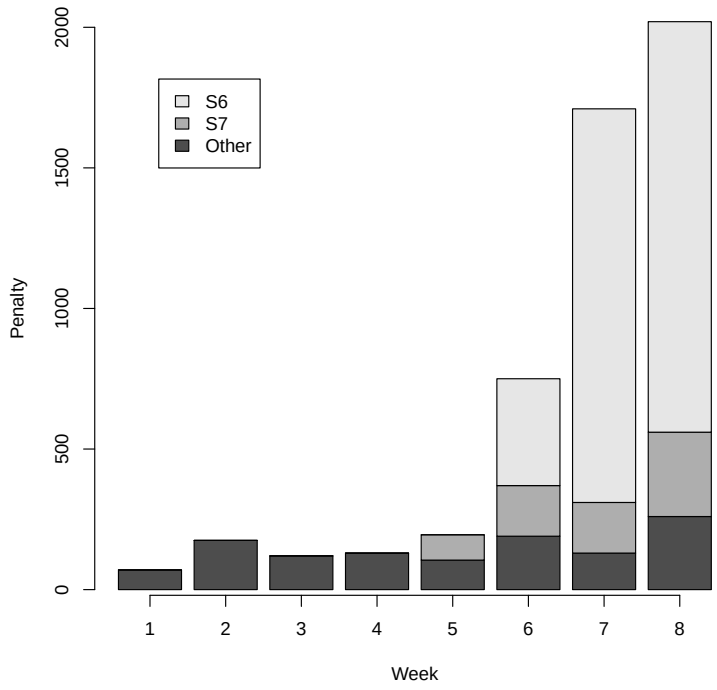

Figure 5.1: Distribution of the penalties incurred by the eight stages of the instance n035w8_2_9-7-2-2-5-7-4-3.

5.2.1 Overstaffing

If one looks at the total number of shifts for each nurse, it becomes apparent that nearly all nurses exceed their maximum number of assignments (compare Figure 5.2), except for nurses with a full time contract. Even these nurses mostly have all their available shifts assigned.

This is despite the fact that, ignoring the other constraints, the total number of assignments available is easily sufficient to cover all shifts at the optimal level (1180 available versus 1029 needed for optimal staffing levels in the example instance).

However, since there is no penalty on overstaffing, there is no pressure to avoid unnecessary assignments. Indeed, in some cases it can seem advantageous to assign shifts above the optimal staffing levels in order to fulfil sequence constraints or the complete weekend constraints (S7).

However, as soon as the available assignments are used up, high penalties are unavoidable as other constraints (in particular cover constraints and sequence constraints) still have to be fulfilled.

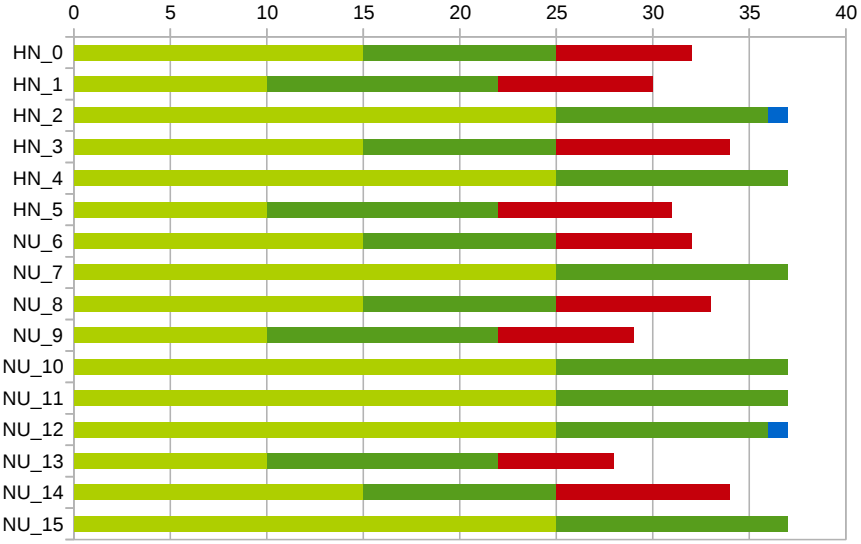

Figure 5.2: Total number of assignments for the first 15 nurses in the solution of the instance n035w8_2_9-7-2-2-5-7-4-3. Red marks assignments exceeding the maximum, blue indicates remaining unassigned shifts below the maximum. The light green part denotes the minimum number of assignments for each nurse.

This can also be seen from Figure 5.3: In earlier weeks, far more shifts are assigned to nurses than necessary, while in later weeks, constraints S6 force solutions to be closer to the required staffing levels.

To avoid this situation, the following constraint is added to the model:

S8*. Overstaffing The number of nurses assigned to each shift and skill per day should not exceed the optimal coverage levels.

This constraint can be added to the basic IP model with the following inequalities:

$$\begin{aligned} \text{S8*} \quad & \forall s \in S, k \in K, d \in \{1 \dots 7\} \\ & \sum_{n \in N} x^d_{nsk} \leq o^d_{sk} + C^{S8*}_{skd} \end{aligned} \tag{5.28}$$

where C^{S8*}_{skd} is a new (non-negative) surplus variable. These new variables need to be included in the objective function too. A discussion about the optimal weights for all artificial constraints can be found in Chapter 7.

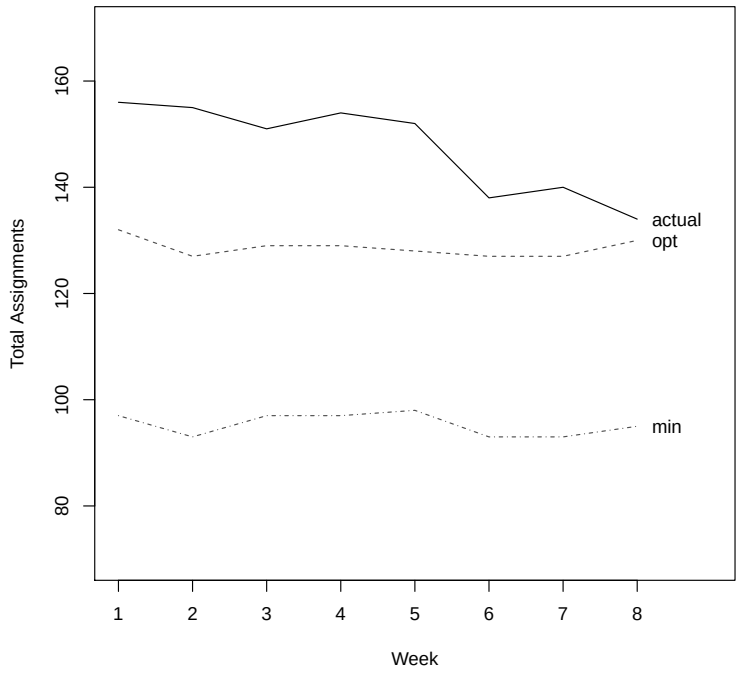

Figure 5.3: Distribution of the total number of assignments in the eight stages of the instance n035w8_2_9-7-2-2-5-7-4-3.

5.2.2 Average assignments

While the overstaffing constraints already reduce the number of excess assignments over the maximum, they do not differentiate between nurses with different contracts. As a result, nurses with part-time or half-time contracts have the same schedules as those with full-time contracts in the earlier weeks. Consequently, they are not available in later weeks without penalties, as their contracts are already maxed out.

Ideally, nurses should be employed according to their contracts during all weeks, with full-time nurses having more assignments per week than other nurses. In part this is already done implicitly, because nurses with shorter contracts usually also have shorter work stretch lengths and longer rest stretch lengths.

To ensure that each nurse will be available until the last stage, their assignments should be distributed evenly across all stages.

S6*. Average assignments The total number of assignments up to the current week must be within the bounds defined in the contract, multiplied by the fraction of weeks that have already passed.

This constraint generalizes constraints S6 to earlier weeks.

To give an example, if a nurse has a minimum of 10 assignments and a maximum of 22, then after stage 4 (of 8), they should have between 5 and 11 shifts assigned. Assuming that they already had 7 shifts assigned in stages 1 to 3, this constraint would require them to have between 0 and 4 assignments in stage 4.

If these constraints are satisfied during all weeks, it can be guaranteed that also constraints S6 are satisfied for the whole schedule.

The following inequalities model these constraints:

$$\text{S6*} \quad \forall n \in N$$

$$a_n^{tot} + \sum_{\substack{s \in S \\ k \in K \\ d \in \{1...7\}}} x_{nsk}^d \leq \lfloor a_n^+ * \frac{w}{w^+} \rfloor \; + \; C_n^{S6*} \tag{5.29}$$

$$\forall n \in N$$

$$a_n^{tot} + \sum_{\substack{s \in S \\ k \in K \\ d \in \{1...7\}}} x_{nsk}^d \geq \lceil a_n^- * \frac{w}{w^+} \rceil \; - \; C_n^{S6*} \tag{5.30}$$

C_n^{S6*} are again new surplus variables. As at most one of the inequalities without surplus variables can be violated at the same time, it is possible to reuse the surplus variables for both equations.

Here, fractional limits are rounded such that the limits are always integer numbers and the solutions satisfying S6* always also satisfy S6. However, different rounding schemes did not influence the quality of the generated solutions.

An alternative version of constraints S6* can be formulated as

S6*b. Average assignments In each week, the remaining assignments (not yet used in previous weeks) should be divided equally among all remaining weeks.

Continuing the example above, the nurse in question would have between 3 and 15 assignments left to distribute over 5 weeks (stages 4 to 8). This means that according to constraint S6*b, they should have between $\frac{3}{5}$ (rounded up to 1) and 3 shifts assigned during stage 4.

$$\text{S6*b} \quad \forall n \in N$$

$$\sum_{\substack{s \in S \\ k \in K \\ d \in \{1...7\}}} x_{nsk}^d \leq \lfloor (a_n^+ - a_n^{tot}) * \frac{1}{w^+ - w + 1} \rfloor \; + \; C_n^{S6*} \tag{5.31}$$

$$\forall n \in N$$
$$\sum_{\substack{s\in S\\ k\in K\\ d\in\{1...7\}}} x^d_{nsk} \geq \lceil (a^-_n - a^{tot}_n) * \frac{1}{w^+ - w + 1} \rceil \; - \; C^{S6*}_n \tag{5.32}$$

The difference between these two formulations becomes visible in case of an imbalance in preceding stages (i.e. too many or too few assignments): S6* tries to restore the balance (which might require unusual work or rest stretches), while S6*b ignores a global balance and works exclusively with the assignments remaining for the current and future stages. A further discussion of these two formulations can be found in Chapter 7.

5.2.3 Average working weekends

The same argument as above also applies to constraints S7, the maximum number of total weekends. Just like assignments in general, also weekends should not be used up in the early stages, but distributed across all weeks to preserve options.

Therefore, an analoguous constraint S7* can be defined:

S7*. Average working weekends In each week, the still available working weekends (not yet used in previous weeks) should be divided equally among all remaining weeks.

with the corresponding inequalities

$$\text{S7*} \quad \forall n \in N$$
$$W_n \leq \lfloor (t^+_n - t^{tot}_n) * \frac{1}{|W| - w + 1} \rfloor \; + \; C^{S7*}_n \tag{5.33}$$

Note that since there is at most one working weekend per week and nurse, and the maximum number of working weekends is less than the number of weeks, the limit set for each week will either be 0 or 1.

5.2.4 Next week restrictions

In addition to the global constraints, solutions for different stages influence each other also at the boundary between weeks.

Since the staffing requirements for the next week are unknown in each stage, leaving more options to schedule nurses without conflicts is beneficial. If there are only few good assignments for the nurses with a certain skill, satisfying the cover constraints might become difficult if they do not match one of those options.

A common way for schedules to restrict the options for the next stage is via the sequence constraints. For example, let the minimum number of consecutive night shifts (σ_{N}^{-}) be 4 and the proposed solution for this week end with a single night shift on Sunday for a nurse (and any other shift or a day off on Saturday, compare Figure 5.4). Then we already know that any assignment for this nurse from Monday to Wednesday that is not a night shift, will inevitably incur a penalty (and depending on the rest of the schedule, assigning only night shifts on these three days could result in penalties of its own).

As another example, if the maximum number of consecutive night shifts is 5 and the proposed solution already contains a shift stretch of at least 5 night shifts in the days leading up to Sunday, this means that assigning a further night shift on Monday of the next week would incur a penalty for exceeding the maximum length.

	Sa	Su	Mo	Tu	We	
...	-	N	N?	N?	N?	...

Figure 5.4: Assignment that heavily restricts options for the following week. Assuming $\sigma_{\mathrm{N}}^{-} = 4$, a single night shift on Sunday will cause a penalty in the next week if any shifts other than additional night shifts have to be assigned between Monday and Wednesday.

The same reasoning applies to work and rest stretches.

S9*. Restriction of next week's assignments Options for next week's schedule should not be restricted. The penalty is calculated as the total number of shifts that cannot be assigned in the next week without violating at least one sequence constraint.

The equations to model this constraint are split into restrictions from shift (a), work (b) and rest (c) stretches, each regarding the minimum and maximum stretch length and with their own set of surplus variables.

S9*a $\quad \forall n \in N, s \in S, b \in \{1 \dots (\sigma_s^- - 1)\}$

$$\sum_{k \in K} ((1 - x_{nsk}^{7-b}) + \sum_{i \in \{0 \dots (b-1)\}} x_{nsk}^{7-i}) \leq b + \frac{C_n^{S9*a}}{|S|(\sigma_s^- - b)} \tag{5.34}$$

$\forall n \in N, s \in S$

$$\sum_{k \in K} \sum_{i \in \{0 \dots (\sigma_s^+ - 1)\}} x_{nsk}^{7-i} \leq \sigma_s^+ - 1 + C_n^{S9*a} \tag{5.35}$$

S9*b $\quad \forall n \in N, b \in \{1 \dots (w_n^- - 1)\}$

$$\sum_{\substack{s \in S \\ k \in K}} ((1 - x_{nsk}^{7-b}) + \sum_{i \in \{0 \dots (b-1)\}} x_{nsk}^{7-i}) \leq b + \frac{C_n^{S9*b}}{w_n^- - b} \tag{5.36}$$

$$\forall n \in N$$

$$\sum_{\substack{s \in S \\ k \in K}} \sum_{i \in \{0 \dots (w_n^+ - 1)\}} x_{nsk}^{7-i} \leq w_n^+ - 1 + \frac{C_n^{S9*b}}{|S|} \tag{5.37}$$

S9*c $\quad \forall n \in N, b \in \{1 \dots (f_n^- - 1)\}$

$$\sum_{\substack{s \in S \\ k \in K}} (x_{nsk}^{7-b} - \sum_{i \in \{0 \dots (b-1)\}} x_{nsk}^{7-i}) \leq 0 + \frac{C_n^{S9*c}}{|S|(w_n^- - b)} \tag{5.38}$$

$$\forall n \in N$$

$$\sum_{\substack{s \in S \\ k \in K}} \sum_{i \in \{0 \dots (f_n^+ - 1)\}} x_{nsk}^{7-i} \geq 1 - C_n^{S9*c} \tag{5.39}$$

5.2.5 Unresolvable patterns

In the solutions generated for various instances, violations of sequence constraints most commonly appear at the boundaries between weeks. In many cases, this is the result of patterns similar to those shown in Figure 5.5.

	Mo	Tu	We	Th	Fr	Sa	Su	Mo	
N_1	-	-	D	D	N	N	N	?	...

Figure 5.5: Assuming that $\sigma_{\mathrm{N}}^- = 4$ and $w_{N_1}^+ = 5$, the maximum work stretch length is already reached but at least one more night shift at the beginning of the next week is required.

In general, not checking the feasibility of completing a multi-shift work stretch in the next week can lead to situations where the last shift stretch can not be extended to the minimum length without violating the maximum work stretch length.

This leads to the following additional constraint:

S10*. Unresolvable Patterns For work stretches at the end of the week, there should be a way to complete them in the next week without violating either the maximum work stretch length or the minimum shift stretch length.

Assume a stretch of shift s is assigned to nurse n at the end of the week. Then an unresolvable pattern has the following structure: First, a block of at least $w_n^+ - \sigma_s^-$ shifts (that can be any type except a day off) is scheduled (A), followed by a single shift that is not s (B). Then, the remaining b days up to the end of the week are filled with assignments to shift s (C), where $b < \sigma_s^-$.

To avoid a violation of the minimum shift stretch length, at least $\sigma_s^- - b$ more days of shift s would be required at the start of the next week. However, together with parts (A) and

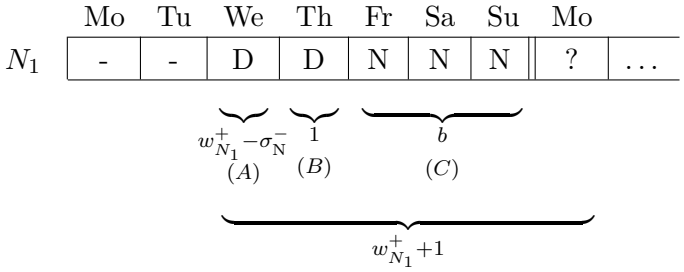

Figure 5.6: The same pattern as for Figure 5.5, split up into the parts matched by the constraints S10*. For this assignment, $b = 3$.

(B), this would bring the total work stretch length to at least $(w_n^+ - \sigma_s^-) + 1 + b + (\sigma_s^- - b) = w_n^+ + 1$, which exceeds the maximum work stretch length w_n^+. The different parts are visualized in Figure 5.6.

Equations 5.40 detect and penalize these patterns through the use of a further set of surplus variables.

$$\begin{aligned} \text{S10*} \quad & \forall n \in N, s \in S, b \in \{1 \dots \sigma_s^- - 1\} \\ & \sum_{k \in K} \Big(\overbrace{\sum_{\substack{j \in \{1 \dots w_n^+ - \sigma_s^-\} \\ t \in S}} x_{ntk}^{7-b-j}}^{\text{(A)}} + \overbrace{\sum_{t \in S \setminus s} x_{ntk}^{7-b}}^{\text{(B)}} + \overbrace{\sum_{i \in \{0 \dots b-1\}} x_{nsk}^{7-i}}^{\text{(C)}} \Big) \\ & \leq w_n^+ - (\sigma_s^- - b) + C_n^{S10*} \end{aligned} \tag{5.40}$$

5.2.6 Objective Function

To have an impact on the generated solutions, the surplus variables for the added constraints have to be included in the objective function. The objective function f' for the extended model is therefore

$$\begin{aligned} \textit{minimize } f' = \quad & f \\ & + W^{S8*} * \sum_{s \in S} \sum_{k \in K} \sum_{d \in \{1 \dots 7\}} C_{skd}^{S8*} \\ & + W^{S6*} * \sum_{n \in N} C_n^{S6*} \\ & + W^{S7*} * \sum_{n \in N} C_n^{S7*} \\ & + W^{S9*} * \sum_{n \in N} (C_n^{S9*a} + C_n^{S9*b} + C_n^{S9*c}) \\ & + W^{S10*} * \sum_{n \in N} C_n^{S10*} \end{aligned}$$

where W^{S8*}, W^{S6*}, W^{S7*}, W^{S9*} and W^{S10*} are the weights of their corresponding constraints.

CHAPTER 6

Local Search

To compare results with the IP model described above, a local search framework was implemented for this problem. Similar to the approach described in [Mus06], a combination of the Tabu-Search (TS) [Glo89] and Min-Conflicts (MC) [MJPL92] heuristics was used: Starting from an initial solution, at each step, with probability p, one move of the TS strategy is applied, and with probability $1 - p$, one move of the MC strategy is applied. After a fixed number of successive moves that did not improve the current best solution, the search is restarted from a new initial solution. The complete algorithm is described in Algorithm 6.1.

Algorithm 6.1: Local Search

```
1  while ¬ timeout do
2  |  GenerateInitialSolution();
3  |  while ¬ max number non-improving moves reached do
4  |  |  if random() < p then
5  |  |  |  Apply move of TabuSearch strategy;
6  |  |  else
7  |  |  |  Apply move of MinConflict strategy;
8  |  |  end
9  |  |  if F(currentSolution) < F(bestSolution) then
10 |  |  |  bestSolution ← currentSolution;
11 |  |  end
12 |  end
13 end
14 return bestSolution
```

For both heuristics, a neighbourhood composed of two types of moves was employed:

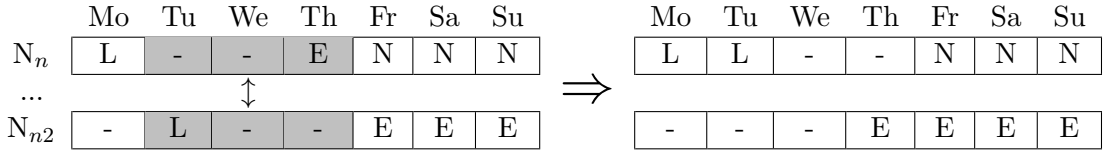

Figure 6.1: Example of a *swap* move between nurses n and $n2$, starting on Tuesday ($d = 2$), with a length of $l = 3$ (assigned skills are not shown).

change(n, d, s, k) sets the shift and skill assignment of nurse n on day d to s and k, respectively. If $s = 0$, this is treated as a day off. Only assignments to a legal skill for this nurse are allowed.

swap$(n, d, n2, l)$ swaps the assignments of the next l days, starting at day d, of nurse n with those of nurse $n2$. As before, only moves that don't result in conflicts with the skills of both nurses are allowed. Blocks up to length $l = 4$ are considered in each step.

An example of a *swap* move that could be used to resolve both shift and rest stretch length conflicts is shown in Figure 6.1.

In the problem treated in [Mus06], the exact number of employees for each day and shift was known in advance, making *change* moves unnecessary, as all staffing constraints could already be fulfilled by the initial solution. This approach is infeasible for the current problem, for multiple reasons: Since varying the coverage of a particular shift between the minimum and optimum levels is only a soft constraint violation and going beyond the optimal coverage not even that, the number of nurses assigned to this shift in a schedule must be left variable. Further, the presence of skill restrictions (with nurses often being qualified for multiple skills) and hard constraints concerning shift successions (H3) makes it hard to construct even a feasible initial solution.

For this reason, initial solutions were not required to fulfil hard constraints H2 (Minimal staffing) and H3 (Shift type successions), which were converted to soft constraints with a sufficiently large weight (3000 for H2 and 1000 for H3). While feasibility of the final solution cannot be guaranteed using this approach, all implemented heuristics were able to produce a feasible solution within only a few steps.

For the generation of an initial solution, multiple approaches where implemented.

Random assignment Each nurse on each day is assigned a random shift and a random (valid) skill, with fixed probabilities for days off.

Randomised greedy Shifts up to optimum coverage (for each day and skill) are assigned in a random order to the nurse resulting in the lowest objective value. It is possible that already assigned shifts are overwritten by later assignments, though it is unlikely that the minimum staffing levels are violated, given the high weight of these constraints.

Heuristic ordering Similar to above, except that assignments are ordered according to their importance and expected difficulty to assign. Such an approach is detailed in [BCP+08], although the criteria and their weights are adjusted to the current problem. Specifically, assignments up to the minimum coverage are listed first, with priority being given to night shifts, shifts at the start and end of the week and skills that only few nurses are qualified for.

The more sophisticated algorithms usually produced schedules with only few violations of hard constraints that could be made feasible in the first few steps of the local search part. However, this came at the cost of a higher complexity and time needed, and had no noticeable impact on the quality of the final solutions. Ultimately, the random assignment approach was used, with an additional check to avoid assigning shifts over and above the optimal coverage level.

To reduce the time needed to arrive at a feasible solution, only *change* moves are considered until the first feasible solution has been found.

Also a hybrid form of TS and MC (MC-T) was experimented with, where the MC heuristic additionally made use of a tabu list, but this approach did not result in good solutions.

As an alternative to the MC heuristic, also Random Walk was implemented, where a random move is chosen among all those involving at least one conflict. During the experiments, it turned out that the speed gained from replacing MC by a simpler heuristic did not make up for the additional noise introduced into the solutions.

Besides, the main bottleneck lies in the TS part, where for each step, all possible moves have to be evaluated for their objective value. To speed up this process, the evaluation function makes heavy use of caching and the fact that a single *swap* move influences the schedule of at most two nurses (one for *change* moves) and staffing levels can change at most for one day with *change* moves and not at all for *swap* moves. After each move, the quality of each nurse's schedule as well as the coverage constraint violations of each day are evaluated separately and stored in a cache. During the evaluation of a candidate move, only the changed parts are reevaluated, while the quality of all other parts is taken directly from the cache.

CHAPTER 7

Experimental Results

Both the IP formulations and the LS framework were implemented in Java 7, using the IBM ILOG CPLEX solver[1], version 12.6.3, to solve the IP models. All experiments were performed on an Intel Xeon 2.33GHz PC, using a single thread. The time limit for each week was set to the time allotted by the benchmarking script[2] provided for the INRC-II (see Table 7.1).

Nurses	Time (s)
30	69.54
35	95.62
40	121.70
50	173.86
60	226.02
70	278.18
80	330.34
100	434.65
110	486.81
120	538.97

Table 7.1: Time limits allotted by the benchmarking script of the INRC-II for instances of varying sizes

As data set for tuning parameters, the set of late instances[3] published for the INRC-II was used. The models were evaluated on the set of hidden instances [4].

[1] http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/
[2] http://mobiz.vives.be/inrc2/?page_id=245
[3] http://mobiz.vives.be/inrc2/wp-content/uploads/2014/08/late-instances.txt
[4] http://mobiz.vives.be/inrc2/wp-content/uploads/2014/08/hidden-instances.txt

7.1 Model Extensions

The results given in this section are for the IP formulations, those of the LS framework are discussed in Section 7.2.

Figure 7.1 shows the effects of adding the individual extensions described in Section 5.2 to the basic model. Due to the similarity in structure and purpose, constraints S6* (Average Assignments) and S7* (Average Weekends) are used together. For this comparison, the weight of each additional constraint was set to a value of 1 to ensure that the focus of the optimization still remains on the original constraints. The only exception is constraint S10* (Unresolvable patterns), since a violation of this constraint directly results in a violation of at least one shift stretch length constraint in the next week and thus warrants a weight of 15 (as if the violation had already occured). These weights will be referenced as *standard weights*.

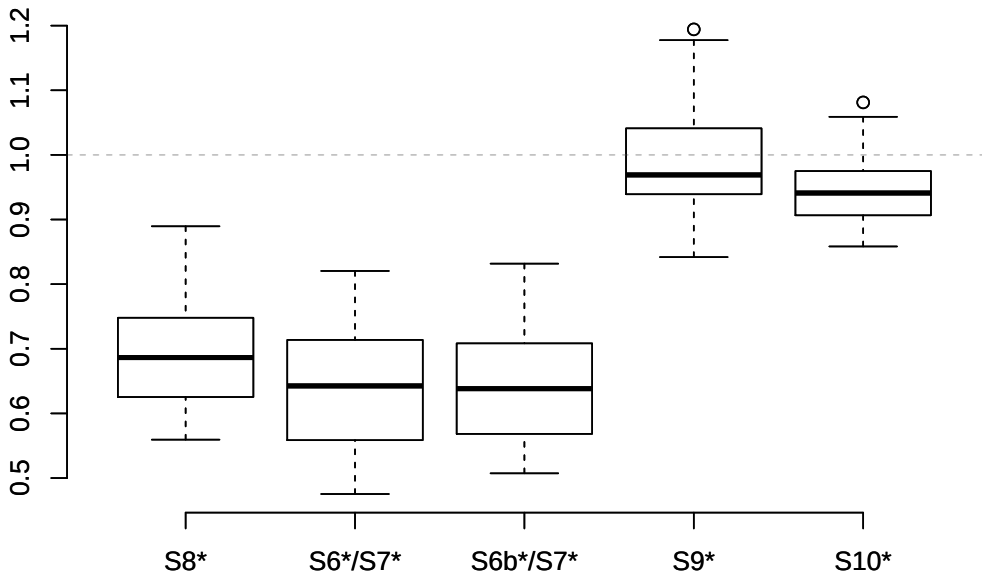

Figure 7.1: Performance of the basic model extended by each set of constraints individually. The baseline (value of 1) for each instance is the solution generated by the basic model.

It can be seen that especially the constraints addressing the global assignment constraints (S6* (either version) and S8* (Overstaffing)) result in large improvements to the quality of the solution, in the most extreme cases halving the penalty incurred with the basic

model. Constraints S9* (Next Week Restriction) and S10* also slightly improve upon the solutions of the basic model.

The penalty can be further decreased by combining multiple extensions. During the experiments, it turned out that the constraints S6* (and S7*), S8* and S10* work well together, each of them further increasing the quality of the solutions. This is shown in Figure 7.2: The results of the models using only 2 of the 3 extensions are nearly always worse than in the case where all 3 extensions are used. The models are also better than the basic model or those with only a single extension.

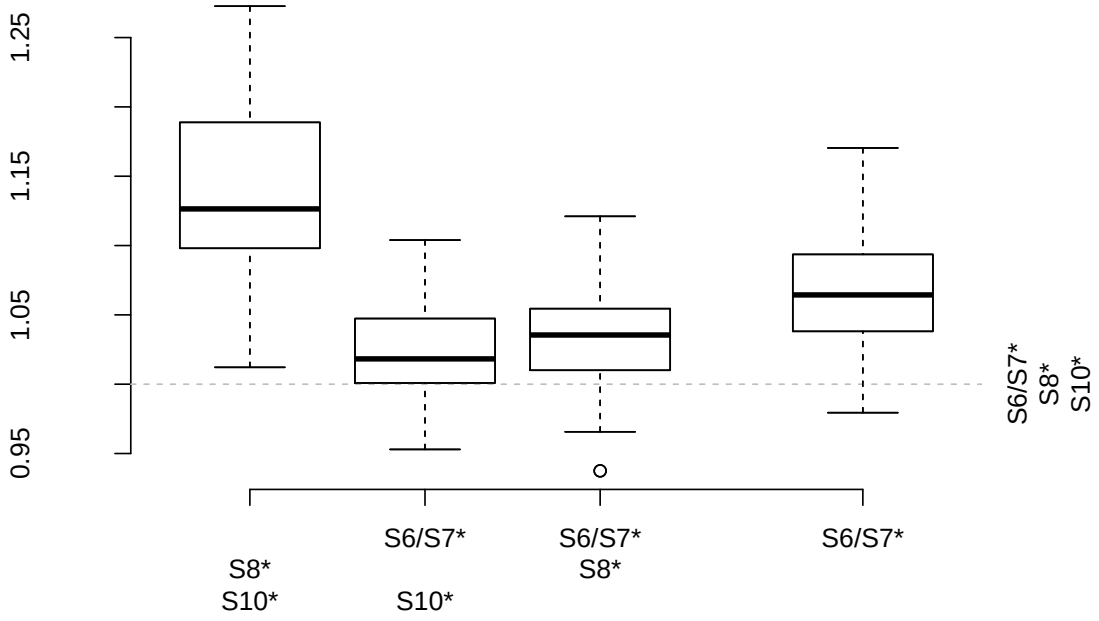

Figure 7.2: Performance of the basic model extended by multiple additional constraints. The baseline (value of 1) for each instance is the solution generated by the extended model with all 3 constraints. For comparison, the results for the model with only constraints S6* (/S7*) from Figure 7.1 are shown too.

Constraints S9* did not combine well with the other constraints, and the results are worse than those without S9*, by between 20% and 50% for most instances. This remained the case even if S9* was assigned a much smaller weight than the other constraints (compare Figure 7.3). Assigning a higher weight to S9* produced even worse results.

For all further experiments, the evaluated models use the extensions S6*, S7*, S8* and S10*, but not S9*. This will be denoted as the *extended model*.

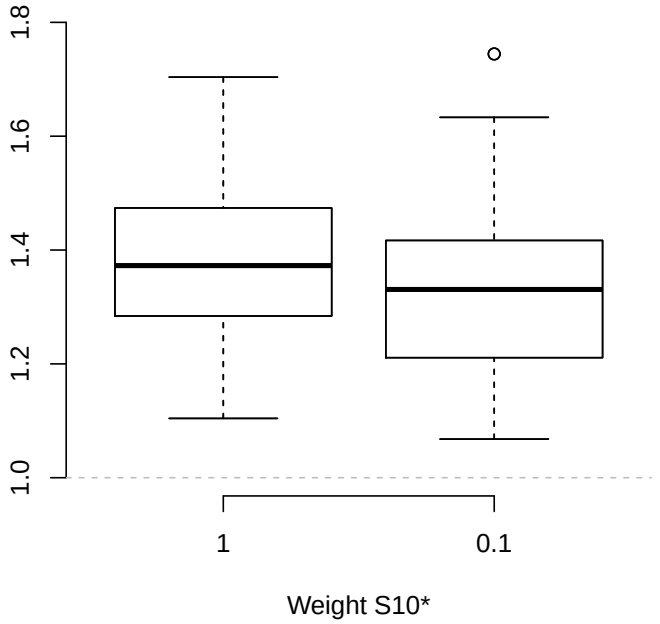

Figure 7.3: Comparison of performance of the extended model (S6*, S7*, S8*, S10*, at standard weights) combined with S9* at weights 1 and 0.1. As before, the baseline value of 1 indicates the solution quality achieved with the extended model without S9*.

Considering the two variants of S6*, S6* produces slights better results than S6*b, but the differences are marginal (see Figure 7.4).

7.1.1 Extension weights

The standard weights $W^?$ of the constraints used so far were chosen more or less arbitrarily (with the exception of S10*, which directly represent a penalty that is sure to appear in the next week). During the experiments it turned out that assigning weights below 1 had little influence on the quality of solutions. This is not surprising given that at weight 1, the original constraints already trump the extensions in importance and the solver mostly optimizes against the basic model and considers the extensions only to distinguish between multiple equivalent solutions. Conversely, assigning weights greater than 1 to some artificial constraints did influence the solutions that were produced, as the relative importance compared to the original constraints and the other extensions was changed. However, if the weights for the extensions were set to values significantly higher than the weights of the original constraints, the results rapidly became worse even than those of the basic model.

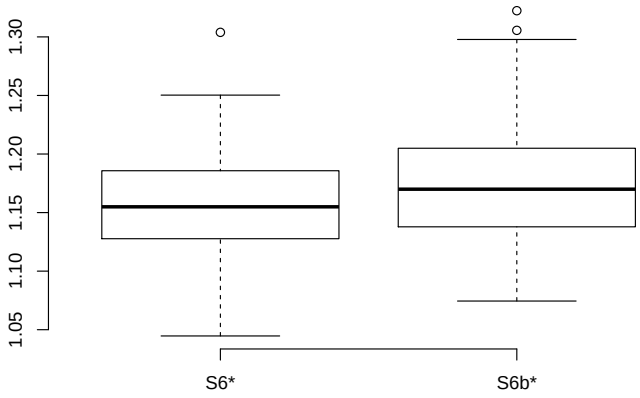

Figure 7.4: Comparison of performance of S6* in both variants. The extended model with standard weights was used. All results were scaled by the best result achieved in any of the models (a value of 1 corresponds to the best result for a certain instance).

#	W^{S6*}	W^{S8*}
1	9.9	11.9
2	10.2	10.2
3	10.0	10.6

Table 7.2: Elite candidates found by IRACE for the values of parameters W^{S6*} and W^{S8*}

To find optimal values for the weights of constraints S6* (/S7*) and S8*, the parameter-tuning framework IRACE [LIDLSB11] was used. Both $W^{S6*}(= W^{S7*})$ and W^{S8*} were varied between 0 and 20, with a precision of one significant digit after the decimal point. As tuning instances, the set of late instances of the INRC-II was used, as mentioned above. IRACE was run in parallel on 4 cores with a limit of 5000 iterations.

The best values for W^{S6*} and W^{S8*} found by IRACE can be seen on Table 7.2. All values are very similar to each other and further experiments showed that the results do not vary significantly under small variations of these values.

7.1.2 Model complexity

Even the direct implementation of all constraints in the basic model, without any particular regard for efficient formulations, could be optimally solved within the time limit for most instances. As can be seen on Table 7.3, even those weeks for which a (provably) optimal solution could not be found, the optimality gap was only at 1.19% on

Model	Optimal	Feasible	Gap (%)	Time (%)
Basic	309	51	1.19	28.4
Extended	274	86	1.90	47.3
Extended+S9*	98	262	7.35	88.0

Table 7.3: Complexity (in terms of time needed to solve each week) of different models. The first two columns give the number of weeks solved to optimality or feasibility (out of 360 in the set of hidden instances) within the time limit. *Gap* is the average optimality gap over all weeks that could not be solved optimally. *Time* is the average time taken by the solver, regardless of whether an optimal solution could be found, relative to the time limit of each instance.

average. Further, stages where usually solved within only slightly more than a quarter of the time available.

The results for the extended model are similar, although the increased complexity of the model had a noticeable effect on the time required to solve each stage. Still, an optimal solution for most weeks could be found, and the optimality gap in the remaining cases still remains below 2%.

When also constraints S9* are added to the model, suddenly only a small fraction of all stages can be solved to optimality, and the average time needed by CPLEX nearly doubles. Also the optimality gap remaining for unsolved stages is much larger than for the extended model alone. This could be one reason why constraints S9* performed so poorly. However, it can not be the only factor, as many solutions using models including S9* were worse than those of other models even in cases where both could be solved optimally.

All this indicates that major improvements can not be expected by more efficient solution strategies and formulations, but will require further extension and refinement of the model. The exception to this are constraints S9*, where a more detailed study of their structure and impact on the size of the model might well result in substantially better performance.

7.2 Local Search

Given the results above, the extended model was also used in the local search framework, with the additional constraints being added to the evaluation function. Since the optimization goal remained the same, it is reasonable to assume that the optimal weights for the IP model are also close to optimal for the local search aproach. For this reason, the same weights were reused without an additional round of re-tuning.

However, the local search framework described in Section 6 contains other parameters that have an impact on the solution quality. In particular, these parameters are p (the probability of following the TabuSearch strategy instead of MinConflict at each step), m

Parameter	Type	Min	Max
p	real	0	0.9
m	integer	10	10000
l	integer	1	50

Table 7.4: Parameters tuned for the local search framework

Param	Weight
W^{S6*}	9.9
W^{S7*}	9.9
W^{S8*}	11.9
W^{S10*}	15

Table 7.5: Constraint weights for model extensions used for the final evaluations

(the number of non-improving moves before a restart) and l (the length of the tabu list, as a multiple of the number of nurses).

As before for the weights, the IRACE framework was used to tune these parameters. The limits for each parameters are summarised in Table 7.4. As before, the set of late instances was used as training data, with 4 threads and 5000 iterations.

The best values for these parameters found by IRACE are $p = 0.5$, $m = 8000$ and $l = 5$. Further experiments showed that the overall solution quality is very robust against small changes of these values.

Figure 7.5 shows an example run of the framework for a single stage. Only feasible solutions were regarded for the plot. It can be seen that after each restart, feasible solutions were found within very few steps and the solution quality improved rapidly before coming to a standstill as local optima were reached. However, it is also noteworthy that the phase leading to the best solution contains two series of over 5000 moves without improvements. If the search had been set to restart substantially sooner, this solution would not have been found. Also of interest is that very good results (within 10% of the best solution found) were achieved after the first 2500 moves, or about 5% of the total running time.

7.3 Final Results

For the final evaluation, the extended model was used, with weights for the extensions as shown on Table 7.5, both for the IP model solved via CPLEX and the Local Search framework. The exact results can be found on Table 7.6, see also Figure 7.6 for comparison.

Due to the extensions, the penalty incurred by the generated solutions is reduced by about 40% on average for the IP model, in some cases even to less than half the penalty

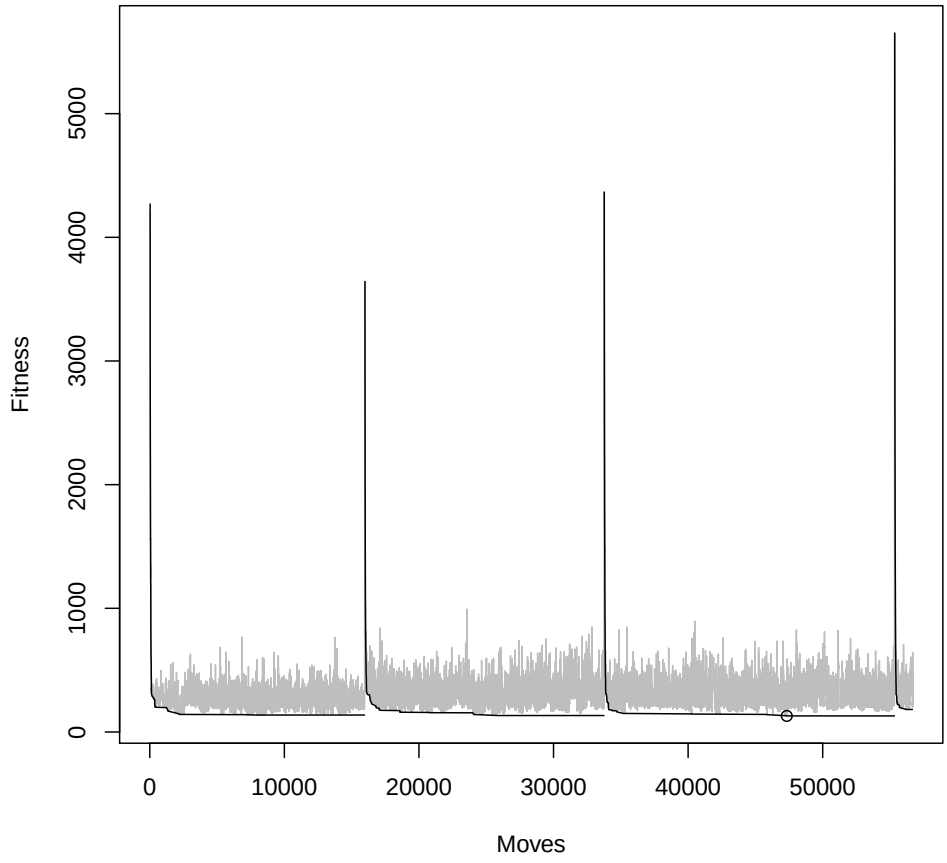

Figure 7.5: Solving the first stage of the instance n035w4_2_8-8-7-5 with the Local Search framework, with the current solution in grey and the best solution within the current search phase in black. Spikes in the fitness value mark points where the search restarted from a new initial solution. Also shown is the move that led to the best overall solution.

of the basic model. Further, there is no instance, where the extended model produced results that were not at least 20% better than those of the basic model.

Also the Local Search framework performed comparatively well, although it was slightly weaker than the IP approach. This is not surprising, considering that most stages could already be solved optimally within the time limit. For larger problems, or with more stringent time limits, it can be expected that LS will outperform any exact approaches.

Compared to the results of the finalists in the INRC-II, the results are competitive (slightly better than the median), although no new best known solutions could be found.

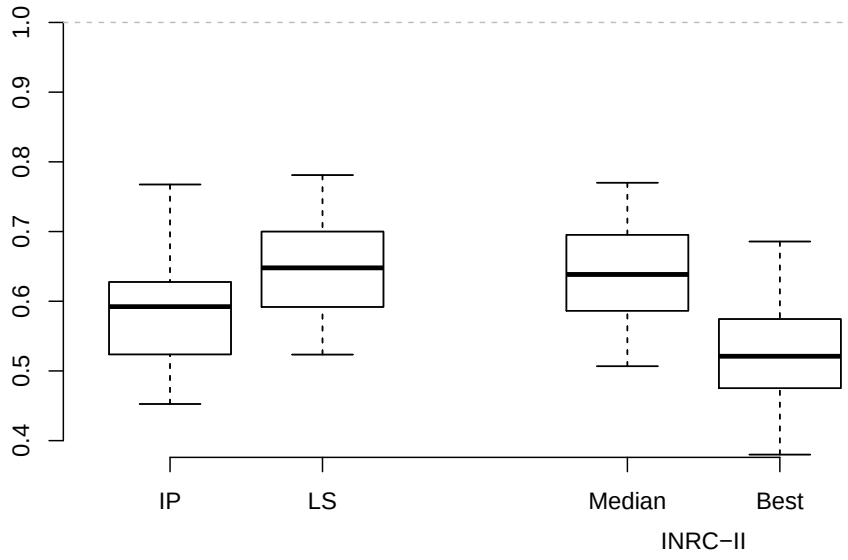

Figure 7.6: Performance of the extended model, both using Integer Programming and Local Search, compared to the solutions produced by the basic model (value of 1). Also shown are the median and best results achieved by the INRC-II finalists.

The average rank over all instances is 3.45, placing these results firmly into the top half of the finalists.

Instance	Basic	Extended	LS	INRC-II Median	INRC-II Best	Rank
n035w4_0_1-7-1-8	2720	1650	1890	1756.5	1630	3
n035w4_0_4-2-1-6	2625	1950	2050	2021.5	1800	3.1
n035w4_0_5-9-5-6	3020	1775	2035	1928.5	1755	2.2
n035w4_0_9-8-7-7	2700	1680	1810	1723.5	1540	4.3
n035w4_1_0-6-9-2	3035	1755	1835	1737	1500	3.3
n035w4_2_8-6-7-1	2495	1645	1605	1644.5	1490	4.5
n035w4_2_8-8-7-5	2375	1410	1420	1407.5	1255	4.2
n035w4_2_9-2-2-6	2675	1950	2050	1947.5	1705	2.7
n035w4_2_9-7-2-2	2645	2030	1950	1970.5	1650	4.1
n035w4_2_9-9-2-1	2700	1840	2040	1927.5	1620	3.3
n035w8_0_6-2-9-8-7-7-9-8	5640	3550	3630	4171	3020	2.1
n035w8_1_0-8-1-6-1-7-2-0	5380	3360	3600	4045.5	2770	3.3
n035w8_1_0-8-4-0-9-1-3-2	5315	3280	3575	4019	2775	3.7
n035w8_1_1-4-4-9-3-5-3-2	5205	3120	3680	3472.5	2805	4.6
n035w8_1_7-0-6-2-1-1-1-6	5795	3370	3535	3548.5	2840	4.1
n035w8_2_2-1-7-1-8-7-4-2	5570	3390	3735	4205	2910	2.5
n035w8_2_7-1-4-9-2-2-6-7	5725	3445	3700	3699.5	2960	3
n035w8_2_8-8-7-5-0-0-6-9	5265	3250	3560	3603	2815	3
n035w8_2_9-5-6-3-9-9-2-1	6040	3515	3880	3659	3045	2.9
n035w8_2_9-7-2-2-5-7-4-3	5340	3155	3680	3508	2715	3
n070w4_0_3-6-5-1	4580	2775	3255	3151	2700	4.3
n070w4_0_4-9-6-7	4030	2545	2895	2889	2430	2.7
n070w4_0_4-9-7-6	4195	2675	3055	2948	2475	3.8
n070w4_0_8-6-0-8	4440	2850	3135	3016	2435	4.1
n070w4_0_9-1-7-5	4010	2665	2950	2864	2320	4.1
n070w4_1_1-3-8-8	4185	2980	3080	3134.5	2700	3.7
n070w4_2_0-5-6-8	4100	2765	2950	3012	2520	4.1
n070w4_2_3-5-8-2	4250	2800	3010	3141.5	2615	3.5
n070w4_2_5-8-2-5	4460	2820	3145	3005.5	2540	4.1
n070w4_2_9-5-6-5	4315	2820	3070	3046	2615	2.5
n070w8_0_3-3-9-2-3-7-5-2	9690	6065	6505	6222	5115	3.5
n070w8_0_9-3-0-7-2-1-1-0	10160	6120	6710	6602	5390	3.3
n070w8_1_5-6-8-5-7-8-5-6	9920	6120	6610	6236.5	5475	3.2
n070w8_1_9-8-9-9-2-8-1-4	9715	5740	6130	6018.5	5100	2.9
n070w8_2_4-9-2-0-2-7-0-6	9995	5660	6635	6259	5410	2.9
n070w8_2_5-1-3-0-8-0-5-8	10310	5810	6695	6315	5280	3.9
n070w8_2_5-7-4-8-7-2-9-9	9885	6010	6855	6317.5	5505	3.9
n070w8_2_6-3-0-1-8-1-5-9	10785	5590	6230	6255	5120	3.6
n070w8_2_8-6-0-1-6-4-7-8	10905	5775	6840	6890.5	5350	3
n070w8_2_9-3-5-2-2-9-2-0	10225	5620	6715	6044.5	5320	2.8
...						

Instance	Basic	Extended	LS	INRC-II Median	INRC-II Best	Rank
...						
n110w4_0_1-4-2-8	6085	2970	3420	3539	2710	4
n110w4_0_1-9-3-5	6110	3185	3750	3663	2920	2.8
n110w4_1_0-1-6-4	6235	3280	3560	4030	2850	3.9
n110w4_1_0-5-8-8	5930	3125	3650	3569.5	2820	3.3
n110w4_1_2-9-2-0	6810	3810	4130	4092	3345	4
n110w4_1_4-8-7-2	6785	3265	3565	3661	2805	3.9
n110w4_2_0-2-7-0	6170	3610	3800	4198.5	3005	3.5
n110w4_2_5-1-3-0	6650	3240	3895	3637.5	2925	4
n110w4_2_8-9-9-2	6725	3990	4285	4025	3415	4
n110w4_2_9-8-4-9	6265	3415	3935	3769	3135	3.3
n110w8_0_2-1-1-7-2-6-4-7	11595	5995	6720	6596	5155	3.9
n110w8_0_3-2-4-9-4-1-3-7	12130	5490	6350	6172.5	4805	4
n110w8_0_5-5-2-2-5-3-4-7	12015	5570	6375	6227	4750	3.8
n110w8_0_7-8-7-5-9-7-8-1	11640	5855	6655	6251.5	4855	3.9
n110w8_0_8-8-0-2-3-4-6-3	11495	5205	6150	6146.5	4465	4
n110w8_0_8-8-2-2-3-2-0-8	12255	5565	6480	6469	4865	3.4
n110w8_1_0-6-1-0-3-2-9-1	12010	5895	6455	6514	5090	3.7
n110w8_1_4-1-3-6-8-8-1-3	11355	5540	6315	6115.5	4315	4
n110w8_2_2-9-5-5-1-8-4-0	12015	5890	6555	6222.5	4770	4
n110w8_2_8-5-7-3-9-8-8-5	11465	5570	6230	5809	4360	3.9

Table 7.6: Results for the basic model, the extended model and the Local Search framework over all instances of the hidden dataset. Added for comparison are the median and best results achieved by the INRC-II finalists for each instance. The last column contains the average rank among the 7 finalists achieved by the extended IP model for each instance (over 10 runs).

CHAPTER 8

Conclusions

This diploma thesis presented both an Integer Programming and a Local Search based approach to solve a nurse rostering problem in a stepping horizon setting. Such settings, where multiple, interdependent stages of a scheduling problem have to be solved consecutively, correspond better to real world practices than the isolated, single stage problems usually studied in academia.

First, a basic IP formulation capable of solving single problem stages was created and used repeatedly for successive stages. This model was adapted to the multi-stage structure by adding new constraints that balance penalties between weeks and favor solutions that don't restrict options for later stages. These constraints are general and modular enough that they can easily be combined and also adapted to different problems with a similar structure.

Using this extended model, substantially improved solutions over those of the original, basic model could be achieved. It also turned out that within the time limits imposed for the INRC-II, most stages could be solved optimally with a state-of-the-art IP solver.

Further, also a LS framework was implemented using the same extensions. While the results did not improve upon those of the exact approach, very good results could be achieved in a fraction of the time, indicating the potential of this approach in tackling larger and more complex variants of the problem.

The results are also competitive with those of the finalists in the INRC-II, where the problem under consideration was first presented, although no new best known solutions could be found.

Potential for future work lies in finding further extensions that improve the existing model. Prioritising nurses according to different qualities like their skills and contracts seems like a good starting point. Also more sophisticated forecasting techniques based

on the requirements of previous weeks could help in adapting the current solution to expected coverage levels of later weeks.

Also similar scheduling problems with a stepping horizon setting could benefit from the extensions developed in this thesis, if they are adapted to the specifics of the problem. Further investigation will be necessary to identify general formulations for repeatedly occuring variants of the new constraints.

Since the time limits used in this thesis were sufficient to solve most of the available instances, not much focus was placed on the efficiency of the IP model. Additional research, possibly including a hybrid approach combined with LS, could enable the same approach to be used for even larger, more complex problems.

List of Figures

List of Tables

List of Algorithms

Bibliography

[BBK09] Jens O Brunner, Jonathan F Bard, and Rainer Kolisch. Flexible shift scheduling of physicians. *Health Care Management Science*, 12(3):285–305, 2009.

[BCBL04] Edmund K. Burke, Patrick De Causmaecker, Greet Vanden Berghe, and Hendrik Van Landeghem. The state of the art of nurse rostering. *J. Scheduling*, 7(6):441–499, 2004.

[BCDCB01] Edmund Burke, Peter Cowling, Patrick De Causmaecker, and Greet Vanden Berghe. A memetic approach to the nurse rostering problem. *Applied Intelligence*, 15(3):199–214, 2001.

[BCP+08] Edmund K. Burke, Timothy Curtois, Gerhard Post, Rong Qu, and Bart Veltman. A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. *European Journal of Operational Research*, 188(2):330 – 341, 2008.

[BLQ10] Edmund K. Burke, Jingpeng Li, and Rong Qu. A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems. *European Journal of Operational Research*, 203(2):484 – 493, 2010.

[BW90] Nagraj Balakrishnan and Richard T. Wong. A network model for the rotating workforce scheduling problem. *Networks*, 20(1):25–42, 1990.

[CG04] Paola Cappanera and Giorgio Gallo. A multicommodity flow approach to the crew rostering problem. *Operations Research*, 52(4):583–596, 2004.

[CGQR11] Marie-Claude Côté, Bernard Gendron, Claude-Guy Quimper, and Louis-Martin Rousseau. Formal languages for integer programming modeling of shift scheduling problems. *Constraints*, 16(1):54–76, 2011.

[CTC+15] Sara Ceschia, Nguyen Dang Thi Thanh, Patrick De Causmaecker, Stefaan Haspeslagh, and Andrea Schaerf. Second international nurse rostering competition (INRC-II) - problem description and rules -. *CoRR*, abs/1501.04177, 2015.

[Dan54] George B. Dantzig. Letter to the editor—a comment on edie's "traffic delays at toll booths". *Journal of the Operations Research Society of America*, 2(3):339–341, 1954.

[Dan02] George B Dantzig. Linear programming. *Operations Research*, 50(1):42–47, 2002.

[dBBB+13] Jorne Van den Bergh, Jeroen Beliën, Philippe De Bruecker, Erik Demeulemeester, and Liesje De Boeck. Personnel scheduling: A literature review. *European Journal of Operational Research*, 226(3):367 – 385, 2013.

[DCVB11] Patrick De Causmaecker and Greet Vanden Berghe. A categorisation of nurse rostering problems. *Journal of Scheduling*, 14(1):3–16, 2011.

[Dow98] Kathryn A Dowsland. Nurse scheduling with tabu search and strategic oscillation. *European journal of operational research*, 106(2):393–407, 1998.

[Edi54] Leslie C Edie. Traffic delays at toll booths. *Journal of the Operations Research Society of America*, 2(2):107–138, 1954.

[EJKS04] A.T 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.

[GK10] Celia A. Glass and Roger A. Knight. The nurse rostering problem: A critical appraisal of the problem structure. *European Journal of Operational Research*, 202(2):379–389, 2010.

[Glo89] Fred Glover. Tabu search—part 1. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[HDCSS14] Stefan Haspeslagh, Patrick De Causmaecker, Andrea Schaerf, and Martin Stølevik. The first international nurse rostering competition 2010. *Annals of Operations Research*, 218(1):221–236, 2014.

[HLT04] G Hao, KK Lai, and M Tan. A neural network application in personnel scheduling. *Annals of Operations Research*, 128(1-4):65–90, 2004.

[KA08] S. Kundu and S. Acharyya. A sat approach for solving the nurse scheduling problem. In *TENCON 2008 - 2008 IEEE Region 10 Conference*, pages 1–6, Nov 2008.

[KG89] Michael M Kostreva and Pierre Genevier. Nurse preferences vs. circadian rhythms in scheduling. *Nursing Management*, 20(7):50–63, 1989.

[KMMA08] S Kundu, M Mahato, B Mahanty, and S Acharyya. Comparative performance of simulated annealing and genetic algorithm in solving nurse scheduling problem. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, pages 96–100, 2008.

[LIDLSB11] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. The irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.

[LLR03] Haibing Li, Andrew Lim, and Brian Rodrigues. A hybrid ai approach for nurse rostering problem. In *Proceedings of the 2003 ACM symposium on Applied computing*, pages 730–735. ACM, 2003.

[MJPL92] Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

[MP04] Margarida Moz and Margarida Vaz Pato. Solving the problem of rerostering nurse schedules with hard constraints: new multicommodity flow models. *Annals of Operations Research*, 128(1-4):179–197, 2004.

[MS98] Andrew J. Mason and Mark C Smith. A nested column generator for solving rostering problems with integer programming. In *International Conference on Optimisation: Techniques and Applications*, pages 827–834, 1998.

[Mus06] Nysret Musliu. Heuristic methods for automatic rotating workforce scheduling. *International Journal of Computational Intelligence Research*, 2(4):309–326, 2006.

[PBVB03] Sanja Petrovic, Gareth Beddoe, and Greet Vanden Berghe. *Storing and Adapting Repair Experiences in Employee Rostering*, pages 148–165. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[PS82] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1982.

[RÏ5] Michael Römer. A multi-commodity flow-based mixed-integer linear programming formulation for nurse rostering problems (abstract). OR2015-International Conference on Operations Research, Vienna, September 2015.

[SFCO15] D. Santos, P. Fernandes, H. Lopes Cardoso, and E. Oliveira. A weighted constraint optimization approach to the nurse scheduling problem. In *Computational Science and Engineering (CSE), 2015 IEEE 18th International Conference on*, pages 233–239, Oct 2015.

[SSV16] Pieter Smet, Fabio Salassa, and Greet Vanden Berghe. Local and global constraint consistency in personnel rostering. Technical report, KU Leuven, 2016.

[STGR14] Haroldo G. Santos, Túlio A. M. Toffolo, Rafael A. M. Gomes, and Sabir Ribas. Integer programming techniques for the nurse rostering problem. *Annals of Operations Research*, pages 1–27, 2014.

[SV12] Fabio Salassa and Greet Vanden Berghe. A stepping horizon view on nurse rostering. In *Proceedings of the 9th international conference on the practice and theory of automated timetabling (PATAT).*, pages 161–173, 2012.