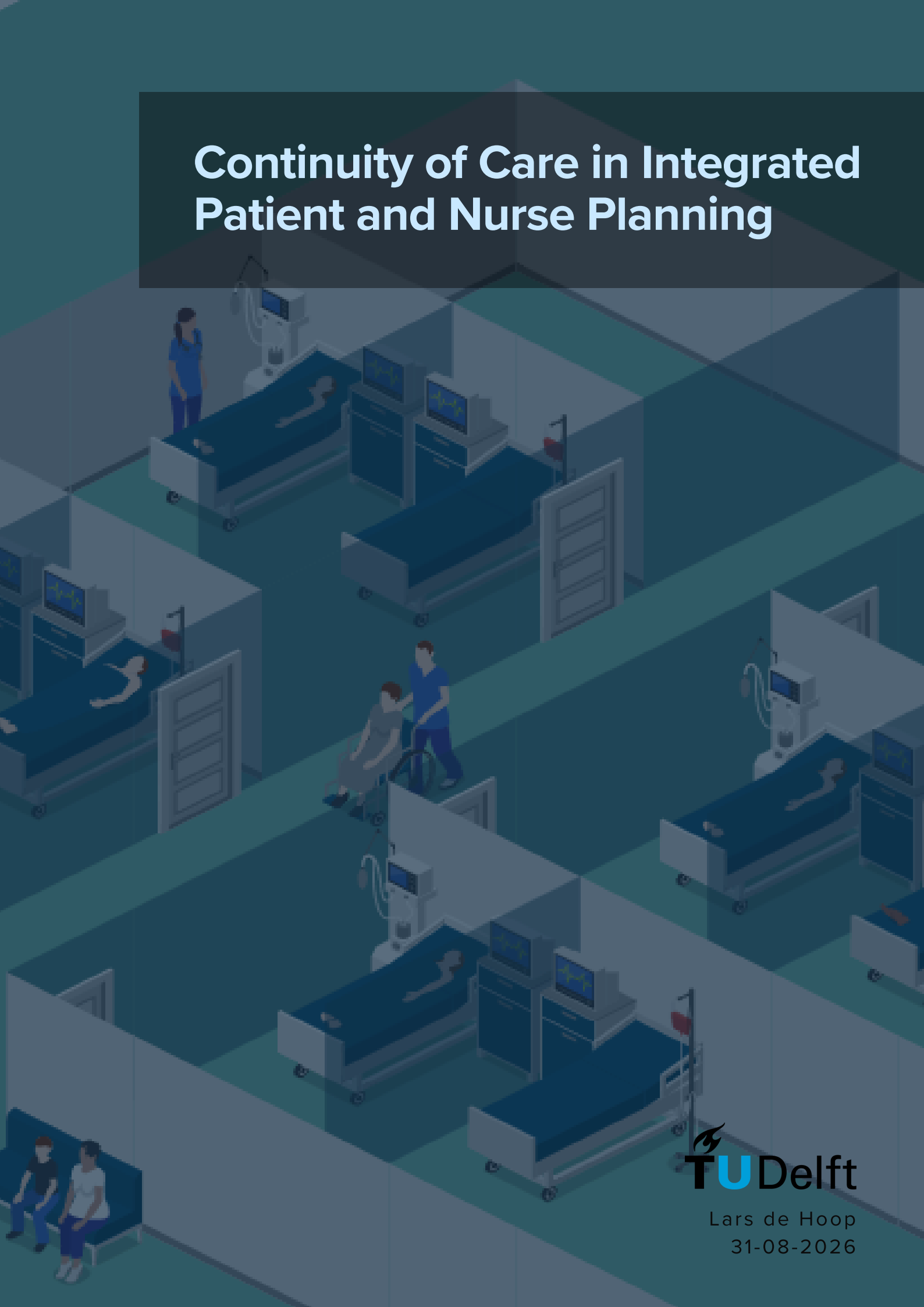
Continuity of Care in Integrated Patient and Nurse Planning



TU Delft
Lars de Hoop
31-08-2026

Continuity of Care in Integrated Patient and Nurse Planning

by

Lars de Hoop

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday August 31, 2026 at 3:00 PM.

Student number:	5644690	
Project duration:	December 2, 2025 – August 31, 2026	
Thesis Committee:	T. van Essen	TU Delft, responsible supervisor
	F. de Meijer	TU Delft, daily supervisor
	W. van Horssen	TU Delft, independent member

An electronic version of this thesis is available at
http://repository.tudelft.nl/.

TU Delft

Acknowledgments

This thesis was conducted within the Discrete Mathematics and Optimization research group of the Delft Institute of Applied Mathematics at Delft University of Technology as part of Master Applied Mathematics. This thesis was written between December 2025 and August 2026.

During the writing of this thesis, some AI tools were used. Claude and ChatGPT were used to help with formatting ILPs in LaTeX and for setting up code to run experiments on DelftBlue.

Firstly, I would like to thank my two supervisors, Frank de Meijer and Theresia van Essen, for their guidance throughout this project. I would also like to thank Wim van Horssen for being part of my graduation committee.

Additionally, I would like to thank my friends and family for supporting me throughout my studies. In particular, I would like to thank Veerle for designing the cover page and I would like to thank Jesse, René and Sanne for proofreading my thesis.

Abstract

The patient-and-nurse-to-room assignment (PNRA) problem is an optimization problem related to hospital wards. This is an integrated problem in which patients and nurses are assigned to rooms in a way that minimizes various objectives. One of these objectives is the continuity of care objective. This objective aims to improve the relationship between a patient and their caregivers by minimizing the number of different nurses a patient has.

Because the PNRA problem is difficult to solve using ILPs, various heuristics are developed that focus on the continuity of care. These are the greedy heuristics, which greedily find a patient-to-room assignment, the interval heuristic, which solves the problem in several intervals, and simulated annealing.

A dataset of 70 instances was used to tune and evaluate the heuristics. These were separated into different sizes based on the number of patients. It was concluded that the performance of the heuristics depends on the instance size. For small instances, the best solutions were found using the interval heuristic and simulated annealing. For larger instances, the greedy non-sequential heuristic found the best solutions.

Contents

Chapter 1

Introduction

Over the last years, the demand in healthcare services has grown. Because of this, a hospital's resources must be carefully distributed in a way that leads to lower expenses, while still ensuring quality of care for patients. Mathematical optimization methods could be employed to more efficiently distribute these resources.

One optimization problem in healthcare is the patient-to-room assignment (PRA) problem. In this problem, patients that have had surgery must be assigned to a room in the hospital ward to recover for a few days. This assignment is subject to capacity constraints and aims to take patient preferences, such as age and gender, into account. Another optimization problem is the nurse-to-patient assignment (NPA) problem, in which nurses are assigned to patients. Here, the goal is often to balance workload across nurses.

The PRA and NRA problems are often considered separately, but in a recent paper by Brandt, Klein, et al. (2025), these two problems were integrated into a single optimization problem. This integrated problem is referred to as the integrated patient-to-room and nurse-to-patient assignment (IPRNPA) problem. Although both problems are already NP-hard, the combination of these two problems is even harder to solve.

In this thesis, a new variation of the IPRNPA problem is considered. In particular, we assume that patients cannot transfer rooms and therefore remain in the same room for the entire duration of their stay. Additionally, we assume that each room can be assigned to only a single nurse. This nurse is responsible for caring for all patients in the room. Nurses are therefore no longer directly assigned to patients, but indirectly based on the room the patient is occupying. This results in more permanent room assignments that have a large impact on which nurses can care for which patients. We refer to this problem as the patient-and-nurse-to-room assignment (PNRA) problem.

Although the PNRA problem considers many different criteria, the main focus is on a single criterion, namely the continuity of care. One way that continuity of care could be interpreted is as a "continuous caring relationship" (Gulliford et al., 2006), in which patients develop a relationship with their caregiver. Higher continuity of care has been shown to reduce mortality rates (Gray et al., 2018), reduce healthcare costs (De Maeseneer et al., 2003) and improve patient satisfaction (Van Walraven et al., 2010). As a result, it is important to account for continuity of care in the nurse-to-patient assignment process. In the PNRA problem, this is done by minimizing

the number of different nurses that are assigned to a patient during their length of stay. However, as this involves decisions across the full planning horizon, this is often difficult to consider.

Additionally, hospitals must make many decisions every day. There are many unpredictable events that could change the schedule: nurses could call in sick, patients' conditions could worsen, surgeries could be delayed or new patients could arrive unexpectedly. In these unpredictable situations, the schedule must be revised. Because of this, we want solution methods that are able to find high-quality solutions in a short amount of time. Moreover, we want these methods to perform well in online settings where the planner has to work with incomplete or non-deterministic data. With this goal in mind, this thesis aims to answer the following questions:

- How can the PNRA problem be modeled using Integer Linear Programming (ILP) and is it able to find high-quality solutions? What ILP models exist for the PNRA problem?
- Which heuristics are most suitable for the PNRA problem and how do they compare in terms of performance and computational time?
- When an online setting is considered, how does the performance of these heuristics change?

These questions are answered in the following chapters. In Chapter 2, relevant literature is reviewed and the problem is formally described. In Chapter 3, the problem is formulated as an ILP and relevant notation is introduced. The problem is further analyzed in Chapter 4, where complexity results are proven, bounds are given and the ILP model is solved. Due to long runtimes for the ILP, several heuristics are introduced in Chapter 5 and tested in Chapter 6. Lastly, in Chapter 7, the problem is extended to an online version in which patients can arrive unexpectedly. The heuristics from Chapter 5 are adapted to and tested on this new setting.

Throughout this thesis, several computational experiments are performed. These are conducted using Python 3.11 and, in the case of ILPs, using Gurobi 13.0.0 (Gurobi Optimization, LLC, 2026) with an academic license. The computations are done using the AMD Ryzen 7 5700U CPU with 16 GB RAM. Additionally, for the tuning of simulated annealing in Chapter 5, TU Delft's DelftBlue (DHPC, 2024) is used. The code used in this thesis can be found on Github[1].

[1] Available at https://github.com/LarsdeHoop/PNRA-problem.

Chapter 2

Preliminaries

In this chapter, the problem is formally introduced. First, relevant literature on the problem is reviewed in Section 2.1. Then, the problem is described in Section 2.2.

2.1 Literature review

Both the nurse-to-patient assignment and the patient-to-room assignment problems have been considered many times in the literature. These are discussed in Sections 2.1.1 and 2.1.2. In Section 2.1.3, the literature on the integration of these two problems and other integrated healthcare problem variations is discussed.

2.1.1 Patient-to-room assignment

The static version of the PRA problem was initially introduced in 2010 by Demeester et al. (2010), although the problem was referred to as the patient admission scheduling (PAS) problem. The problem consists of assigning each patient to a room for each shift of their stay, while minimizing the number of transfers and other preference penalty costs. In this problem description, the only hard constraints are capacity constraints and gender-mixing constraints. The problem is static, since all patient information is known beforehand to the scheduler and is deterministic.

In the literature, the most common solution approaches for static PRA involve local search heuristics. Demeester et al. (2010) use a hybridized tabu search algorithm. Other applied search heuristics are simulated annealing (Ceschia & Schaerf, 2011), an adaptive non-linear great deluge algorithm (Kifah & Abdullah, 2015), the late acceptance hill climbing algorithm (Bolaji et al., 2018), harmony search (Abu Doush et al., 2020) and a biogeography-based evolutionary algorithm (Hammouri, 2022). Apart from these local search heuristics, Borchani et al. (2021) create heuristics based on the Hungarian method, Turhan and Bilgen (2017) apply the MIP-based heuristics fix-and-relax and fix-and-optimize to the problem and Range et al. (2014) use column generation and dynamic constraint aggregation to solve the problem.

In 2012, Ceschia and Schaerf (2012) proposed a dynamic version of the PAS problem in which new patients could unexpectedly arrive during the planning horizon. To account for possible uncertainty in patients' length of stay, an additional "overcrowding risk" objective component was added. This problem was referred to as PASU (PAS with Uncertainty). A simulated annealing algorithm was used

to solve the problem. Another solution approach uses adaptive large neighborhood search (Lusby et al., 2016).

A different dynamic model was proposed by Vancroonenburg et al. (2016). The main difference is that the uncertainty of patient departure dates is now a part of the problem instead of only a part of the objective function. Each patient is given an estimated departure date, which can be different from the true departure date. This dynamic problem is solved using online ILP models.

While the previous dynamic approaches include uncertainty in the model, the objective functions were all completely deterministic. Abera et al. (2020) formulate an ILP with a stochastic objective, taking into account the distribution of the length of stay (LOS) of patients. This problem is solved using a search heuristic with simulated annealing. O'Reilly et al. (2025) also use a stochastic objective and develop a Markov decision process to assist decision making.

Additionally, machine learning methods are used to improve the prediction for certain stochastic model parameters. For example, Farzanegan et al. (2025) try to predict the LOS based on patient information, while Schäfer et al. (2023) estimate emergency patient arrivals.

Other research into the PRA problem includes reducing the number of constraints using constraint aggregation (Liu et al., 2024), proving NP-hardness when minimizing the number of transfers (Brandt et al., 2024) and proving results about feasibility (Brandt, Büsing, & Engelhardt, 2025).

2.1.2 Nurse-to-patient assignment

The first formulation of a nurse-to-patient assignment model was created by Mullinax and Lawley (2002). In the model, nurses are assigned to rooms and to patients in those rooms with the goal of distributing workload evenly across the nurses. The considered objective minimized the difference between the largest and smallest workload in each room. To simplify the problem, the nurses are first assigned to rooms using a first fit decreasing bin packing algorithm, after which the simplified ILP model is solved.

A different method for balancing workload is to minimize the variance of workloads, which is a nonlinear objective. Because of this, solution methods employed mixed integer quadratic programming (Ku et al., 2014) and constraint programming (Schaus & Régin, 2014; Schaus et al., 2009). Another objective function that is sometimes considered is minimizing the number of nurses that are utilized (Marzouk & Kamoun, 2021).

Research into NPA problems is mostly done in the domain of home healthcare, in which nurses must visit patients at home, often considering stochasticity in some form. In this domain, this is also sometimes referred to as the operator-to-patient assignment problem. For example, Lanzarone et al. (2012) assign nurses under various continuity of care requirements while minimizing workload imbalance, considering both deterministic and stochastic demand. In other research, various solution methods are used, including cardinality-constrained robust assignments (Carello & Lanzarone, 2014), stochastic programming (Nguyen et al., 2025), a rolling horizon approach (Güven-Koçak et al., 2024) and an implementer-adversary method (Carello et al., 2024).

To gain better insight into the purpose of the NPA process, Allen (2015) interviewed nurses in charge of the NPA planning. Participants state most often that nurses should be treated in the same way and that workload must be evenly distributed across nurses. A similar study by Van Oostveen et al. (2014) uses the input of focus groups at a hospital to design an NPA optimization model. Applying this model leads to more balanced workloads which also increases staff satisfaction. Similarly, Punnakitikashem et al. (2006) create a prototype for nurse-to-patient assignments, which is tested by nursing students in Baker et al. (2010). Most of the participants state that they would use the prototype to assist in scheduling.

2.1.3 Problem integrations

Only in recent years has the integration of the PRA and NPA problem received attention. This was first done by Brandt, Klein, et al. (2025). Integrating these two problems allows for more complex interactions and can add additional objective function components related to the walking distance and the number of nurses per room. The proposed solution method is a greedy algorithm that also takes similarity of discharge dates into account to assign rooms. An improved solution method by Zanazzo, Ceschia, and Schaerf (2025) consists of a local search heuristic with simulated annealing.

Inspired by this integrated problem, the Integrated Healthcare Timetabling Competition (IHTC) 2024 (Ceschia et al., 2025) considers an expanded version of the problem by Brandt, Klein, et al. (2025) in which the admission date and operating room also have to be determined. Solution methods often involve decomposing the problem into several subproblems. These subproblems are solved using exact/MIP-based methods (Bortoletto et al., 2025; Ciccarelli et al., 2025; Kratz et al., 2025; Rappos et al., 2025), heuristics (Davison et al., 2025; Zanazzo, Smet, et al., 2025) or a combination of the two (Guericke et al., 2025).

Apart from the PRA and NPA problem integration, some other integrated problems are also considered. For example, Ceschia and Schaerf (2012) propose a version of the dynamic PRA problem in which the patient admission date can be delayed by the planner. This problem is also studied by Guido (2024) and solved using a matheuristic called FiNeMath.

Other integrated problems are the integration of a stochastic NPA and the nurse rostering problem (Punnakitikashem et al., 2013) and the integration of the PRA and admission date scheduling (Schmidt et al., 2013).

2.2 Problem description

In this thesis, the patient-to-room and nurse-to-patient assignment problems are integrated into a single problem. This is done by adapting the problem description from the IHTC 2024 (Ceschia et al., 2025). In particular, as we only consider the PRA and NRA problems, the constraints related to surgical case planning are omitted. This simplifies the problem and seems more practical since surgeries are often scheduled further in advance compared to the day-to-day operations of the hospital ward. A similar problem is also considered by Brandt, Klein, et al. (2025), which also does not consider surgical case planning. The changes from these two problem descriptions are highlighted and justified in Section 2.2.4.

As we adapt the formulation from the IHTC, we also use their problem instances. The dataset can be found on their website (Ceschia et al., 2024). This dataset

consists of 70 problem instances with their best known solution. This dataset is split into three categories:

- 10 test instances (labeled test01,..., test10). These were intended for debugging or testing purposes. The optimal solutions were given for these instances.
- 30 public instances (labeled i01,..., i30). The main instances provided for the competition. No optimal solutions were provided for these instances.
- 30 hidden instances (labeled m01,..., m30). These instances were hidden from contestants and used to evaluate their algorithms.

As Chapter 4 shows, solving the PNRA problem using exact methods requires much computation time. For this reason, we consider only the test instances for those exact methods. These test instances contain both very small instances (test03 and test05) and very large instances (test10). For this reason, it gives a good indication as to how problems of different sizes behave. In Chapter 5, the other instances are used to tune and evaluate the heuristics.

Recall that the problem considered in the IHTC also included surgical case planning and that the admission days for patients was therefore also part of the decisions. To adapt this data for our purposes, we use the admission days from the best known solution posted on the website. Apart from this, these solutions are not used during the solving procedure. In case the solutions on the IHTC website updates, the instances used in this thesis can also be found on Github[1].

2.2.1 Assumptions

To make the problem definition more precise, several assumptions must be made.

First, we assume that the problem is static and deterministic. This means that all patient information is known in advance and does not change. Realistically, for example, the length of stay might change when the condition of a patient changes. However, we assume that this cannot happen. We also assume that the nurses working schedule has already been determined and we know in which shifts each nurse is working. In Chapter 7, we no longer assume that the problem is static and instead allow patients to arrive unexpectedly, making it an online problem.

Additionally, we assume that a patient always arrives before the start of the admission day and leaves after the end of the discharge day. This means that a patient is present during each shift on each day of their stay. Additionally, we assume that a patient occupies a room for the full duration of their stay and that they cannot transfer rooms. This assumption is also relaxed in Chapter 7, where transfers are allowed to ensure feasibility.

Finally, we assume that only a single nurse is responsible for each patient during each shift. In reality, it can happen that different nurses divide patient care when shifts overlap. Furthermore, we also assume that all patients in the same room are assigned to the same nurse. This means that we actually assign nurses to rooms instead of to patients directly.

[1] Available at https://github.com/LarsdeHoop/PNRA-problem.

2.2.2 Parameters

A problem instance consists of several components, namely the planning horizon and the sets of nurses, patients and rooms.

The planning horizon consists of several consecutive days. Each day is split into three shifts: an early shift, a late shift and a night shift. It is possible for patients to be admitted during the current planning horizon, but be discharged several days after the planning horizon ended. However, the objective function only takes the assignments in the current planning horizon into account.

The set of nurses consists of all nurses assigned during at least one shift of the planning horizon. For each nurse, the working schedule is known. In each of these scheduled shifts, a nurse has a maximum workload that they can be assigned in that shift. This maximum workload can be different in each shift. Additionally, each nurse has a skill level that is the same in each shift. A nurse's skill level is represented by an integer, such that a nurse with a higher skill level can do more than a nurse with a lower skill level. The number of skill levels can be different in each instance.

The set of patients consists of all patients that are present in the hospital during at least one day of the planning horizon. This also includes patients that were admitted before the current planning horizon and are therefore already assigned a room. For each patient, the admission and the discharge dates are known in advance and deterministic. During each shift of a patient's stay, a minimum skill level requirement is known which indicates how skilled a nurse must be to provide the correct level of care. This skill level requirement can be different in each shift. Additionally, each patient is given a workload value in each shift, indicating how much work or effort it takes a nurse to care for that patient. This value can also be referred to as the acuity and can be different in each shift.

The set of rooms consists of all rooms that patients and nurses can be assigned to. Each room has a maximum capacity which is the number of beds in that room. Since rooms can have different equipment present, it might be that a room is incompatible with some patients. This might be general telemetry equipment (e.g., heart rate, blood pressure, oxygen level) or other equipment such as a defibrillator or medical ventilator.

2.2.3 Constraints

The problem is subject to several constraints, which are either hard constraints or soft constraints. Hard constraints must be respected while soft constraints should be violated as little as possible and as such are part of the objective function. We have the following hard constraints:

- **H1** - *Room capacity*: There cannot be more patients assigned to a room than its capacity for each day.
- **H2** - *Incompatible rooms*: Patients cannot be assigned to incompatible rooms. These are rooms that a patient cannot be assigned to due to the equipment available in that room.
- **H3** - *Room allocation*: For each shift, each occupied room must be allocated to exactly one nurse on duty during that shift.

The following are part of the objective function:

- **S1** - *Continuity of care*: The total number of distinct nurses providing care to a patient during their entire stay should be minimized.
- **S2** - *Gender-mixing*: A room cannot cannot contain both male and female patients at the same time. If this is the case, a penalty is added for each room and each day. How this works for transgender or non-binary people is explained in Section 2.2.4.
- **S3** - *Skill violations*: The minimum skill level a nurse must have to provide the required care for a patient during each shift of their stay should be met. If this is not the case, a penalty is incurred based on the difference in skill level.
- **S4** - *Workload violations*: For each shift, the total workload induced by patients staying in rooms assigned to a nurse should not exceed the maximum workload of the nurse for this shift. If this is not met, a penalty is incurred based on how much this is exceeded.
- **S5** - *Workload imbalance*: The available workload should be evenly distributed across the nurses present in each shift. This is measured by comparing the largest and smallest relative workload in each shift. The relative workload is obtained by dividing the workload with the maximum allowed workload of the nurse in that shift.

2.2.4 Justification

In this section, we justify some of the choices made in the problem assumptions and constraints. We also highlight how this problem is different from the problem in the IHTC (Ceschia et al., 2025) and the problem considered by Brandt, Klein, et al. (2025).

The main focus of this thesis is on continuity of care. It has been shown that higher continuity of care reduces mortality rates (Gray et al., 2018), reduces healthcare costs (De Maeseneer et al., 2003) and improves patient satisfaction (Van Walraven et al., 2010). For this reason, this is an important aspect to include in the problem formulation. As we focus on this specific objective component, we want as little

other objective components as possible, while also ensuring the model remains realistic and useful. Other choices in model assumptions and constraints are largely motivated by this.

There are several objective components that are considered in other variations of the problem, but we decided not to include or formulate in a hard way. The first of these aims to minimize the difference in ages of patients assigned to the same room. Demeester et al. (2010), however, consider a different version where it is formulated as a hard constraint. In particular, patients of certain ages could not be assigned to wards belonging to some departments, such as pediatrics. This could be included in the incompatible rooms constraint (**H2**). This objective component is therefore not included in our version of the problem.

Other problem variations (Brandt, Klein, et al., 2025; Demeester et al., 2010) also often allow transfers in their problem, where the number of transfers is then minimized. Transfers are very disruptive for both patients and staff and should therefore only happen if the problem becomes infeasible. Furthermore, as long as the incompatible rooms constraint (**H2**) is not too restrictive, transfers are not needed to ensure feasibility. Because we are using data from the IHTC, in which transfers were not allowed, they are not needed in our case. We therefore assume that transfers are not allowed. This is, however, something that is revisited in Chapter 7.

Due to the integration of patient-to-room and nurse-to-patient problems, a new objective component could be added to the problem. In Brandt, Klein, et al. (2025), a new objective was introduced which aimed to minimize the number of nurses assigned to a room at the same time. In the PNRA problem, we assume that only a single nurse is assigned to a room and must care for all patients in that room. This seems like a reasonable assumption to simplify the problem. In practice, you might have a singular nurse that is responsible for all patients in a room, while other nurses could help them out if it becomes too busy.

In this thesis, we use a soft version of the gender-mixing constraint (**S2**). In some formulations (Ceschia et al., 2025; Demeester et al., 2010), this is implemented as a hard constraint, where rooms cannot contain both male and female patients at the same time. In practice, however, this is often not the case. In the Netherlands, for example, wards are often mixed-gender (ACCESS NL, 2018). However, this does not mean that the patient's gender is never taken into account during the assignment process. In England, for example, it is the NHS's policy to prevent gender-mixed accommodation as much as possible (National Health Service, 2019). It is only justified when a patient requires highly specialized care or when a patient actively chooses so. The same is true in New South Wales, Australia, with the addition that a patient's admission may not be delayed when accommodation of the same gender is not available (NSW Government, 2022). Although gender-mixing is sometimes unavoidable, some hospitals try to prevent it as much as possible. Because of this, we have opted to formulate the gender-mixing constraint in a soft way.

Note that this constraint has no explicit method for accounting for trans or non-binary people. In the problem instances we use, patients are only labeled male or female. According to NHS's guidelines (National Health Service, 2019), trans and non-binary are accommodated based on their gender presentation and the pronouns they use. Additionally, when a patient's gender is not clear to the staff, they can be asked in what type of accommodation they would be most comfortable. If we follow these guidelines, then a trans or non-binary patient

could be included in the problem instance by labeling them with their gender presentation or preferred type of accommodation.

Another constraint that was included in the problem is the skill violations objective component (**S3**). This was included since it is an important aspect in deciding whether to assign a nurse to a patient. A change that was considered was turning the soft constraint into a hard constraint to reduce the number of soft constraints. However, for many problem instances, there was not always an appropriately skilled nurse present during every shift, which means that the problem would then be infeasible. We also considered formulating the problem as hard as possible. This meant that a patient must be assigned to an appropriately skilled nurse and if this was not possible, they must be assigned to the highest skilled nurse present. However, this often meant that all patients were assigned to a single high skilled nurse, which led to many large violations regarding the maximum workload. Since this would be unrealistic, we opted to keep the skill requirement soft.

Finally, a soft constraint is added that is different from other variations of this problem. According to Allen (2015), the most important aspect of a nurse-to-patient assignment is the equal distribution of workload across nurses. In the IHTC (Ceschia et al., 2025), only the workload violations objective component (**S4**) helps in achieving this equal distribution, but it is not sufficient. During initial tests it was found that the workload was often assigned to a small number of nurses, while ensuring that the maximum workload was not exceeded. This meant that many nurses were also assigned no workload. As this seemed unrealistic, the workload imbalance constraint was added (**S5**). This constraint is a combination of the constraint from Brandt, Klein, et al. (2025), where the workload relative to the maximum workload was considered, and Mullinax and Lawley (2002), where the difference between the largest and smallest workload in each shift was minimized.

Chapter 3

Problem formulation

In Section 3.1, we introduce the notation used for the model input. In Section 3.2, we formulate the model mathematically. We describe the variables, constraints and objective components of the ILP model.

3.1 Model input

Table 3.1 contains a description of all the parameters used in this thesis. Note that all sets related to shifts can be partitioned based on the shift types. This is denoted with the appropriate subscript S_{early}, S_{late} or S_{night}.

Table 3.1: Sets and parameters

Symbol	Description
D	Set of days.
S	Set of shifts.
N	Set of all nurses.
P	Set of all patients (including current occupants).
P_F	Set of all female patients.
P_M	Set of all male patients.
O	Set of current occupants.
R	Set of rooms.
$S(n)$	Set of shifts that nurse $n \in N$ is present.
$S(p)$	Set of shifts that patient $p \in P$ is present.
$S(n,p)$	Set of shifts that both nurse $n \in N$ and patient $p \in P$ are present, i.e., $S(n,p) = S(n) \cap S(p)$.
$s_{\text{adm}}(p)$	First shift of patient p's stay in the current planning horizon.
$N(s)$	Set of nurses present during shift $s \in S$.
$P(s)$	Set of patients present during shift $s \in S$.
$R(p)$	Set of rooms that are compatible for patient $p \in P$.
$C(r)$	Capacity of room $r \in R$.
$r_{\text{prev}}(p)$	Room of current occupant $p \in O$.
$\sigma(n)$	Skill level of nurse $n \in N$.
$\sigma_{\text{req}}(p,s)$	Skill requirement of patient $p \in P$ during shift $s \in S(p)$.
$w(p,s)$	Workload produced by patient $p \in P$ during shift $s \in S(p)$.
$w_{\max}(n,s)$	Maximum workload for nurse $n \in N$ during shift $s \in S(n)$.

3.2 Model formulation

Recall that we assume that each patient is present during all shifts for each day of their stay. This means that the capacity constraints or gender-mixing constraints need to be checked only once on each day. For this reason, these constraints are only enforced in the early shift $s \in S_{\text{early}}$. Similarly, the associated variables are also only defined for $s \in S_{\text{early}}$.

To formulate the problem, we introduce the following binary variables:

$$x_{nrs} = \begin{cases} 1, & \text{if nurse } n \in N \text{ is assigned to room } r \in R \text{ during} \\ & \text{shift } s \in S(n), \\ 0, & \text{otherwise,} \end{cases}$$

$$y_{pr} = \begin{cases} 1, & \text{if patient } p \in P \text{ is assigned to room } r \in R, \\ 0, & \text{otherwise,} \end{cases}$$

$$z_{nps} = \begin{cases} 1, & \text{if nurse } n \in N \text{ is assigned to patient } p \in P \text{ during} \\ & \text{shift } s \in S(n,p), \\ 0, & \text{otherwise,} \end{cases}$$

$$f_{rs} = \begin{cases} 1, & \text{if room } r \in R \text{ contains at least one female patient} \\ & \text{during early shift } s \in S_{\text{early}}, \\ 0, & \text{otherwise,} \end{cases}$$

$$m_{rs} = \begin{cases} 1, & \text{if room } r \in R \text{ contains at least one male patient} \\ & \text{during early shift } s \in S_{\text{early}}, \\ 0, & \text{otherwise,} \end{cases}$$

$$v^{\text{gm}}_{rs} = \begin{cases} 1, & \text{if room } r \in R \text{ contains both male and female} \\ & \text{patients during early shift } s \in S_{\text{early}}, \\ 0, & \text{otherwise,} \end{cases}$$

$$a_{np} = \begin{cases} 1, & \text{if nurse } n \in N \text{ is assigned to patient } p \in P \text{ during} \\ & \text{at least one shift,} \\ 0, & \text{otherwise.} \end{cases}$$

We also have the following non-negative real-valued variables:

- v^{sv}_{ps}: A value indicating the skill level requirement violation for patient $p \in P$ during shift $s \in S(p)$. The value is zero when the assigned nurse has the appropriate skill level and otherwise it is the difference between the skill levels. In particular, if n is the assigned nurse, then

$$v^{\text{sv}}_{ps} = \max\left\{0, \sigma_{\text{req}}(p,s) - \sigma(n)\right\}. \tag{3.1}$$

- v^{wv}_{ns}: A value indicating the maximum workload violation for nurse $n \in N$ during shift $s \in S(n)$. If the assigned workload is lower than the maximum, this value is zero. If the assigned workload is higher, then the value is the amount with which the workload is exceeded. In particular, if $w(n,s)$ is the assigned workload of nurse $n \in N$ during shift $s \in S(n)$, then

$$v^{\text{wv}}_{ns} = \max\left\{0, w(n,s) - w_{\max}(n,s)\right\}. \tag{3.2}$$

- l_s^- and l_s^+: Two values describing the smallest and the largest assigned relative workload of a nurse for each shift $s \in S$. In particular, if $w(n, s)$ is the assigned workload of nurse $n \in N$ during shift $s \in S(n)$, then

$$l_s^- = \min_{n \in N(s)} \frac{w(n,s)}{w_{\max}(n,s)}, \tag{3.3}$$

$$l_s^+ = \max_{n \in N(s)} \frac{w(n,s)}{w_{\max}(n,s)}. \tag{3.4}$$

In the following sections, the constraints of the model are described. For constraints directly related to the objective components, only lower bound constraints are needed. Since we are minimizing, corresponding variables are correctly set to the lower bound. For the variables l_s^- and l_s^+ related to the workload imbalance objective, we are minimizing their difference. Therefore, an upper bound is placed on l_s^-.

3.2.1 Patient-to-room assignment

Each patient $p \in P$ is assigned to exactly one room $r \in R$, i.e.,

$$\sum_{r \in R} y_{pr} = 1, \quad \forall p \in P. \tag{3.5}$$

No room $r \in R$ can have more patients assigned than the capacity $C(r)$ on each day of the planning horizon (**H1**). This is enforced during each early shift $s \in S_{\text{early}}$, i.e.,

$$\sum_{p \in P(s)} y_{pr} \leq C(r), \quad \forall r \in R,\, s \in S_{\text{early}}. \tag{3.6}$$

A patient $p \in P$ cannot be assigned to an incompatible room $r \in R \setminus R(p)$ (**H2**), i.e.,

$$y_{pr} = 0, \quad \forall p \in P,\, r \in R \setminus R(p). \tag{3.7}$$

A patient admitted during a previous planning period (current occupant) $p \in O$ is assigned to the room $r_{\text{prev}}(p)$ they were already assigned to, i.e.,

$$y_{pr_{\text{prev}}(p)} = 1, \quad \forall p \in O. \tag{3.8}$$

3.2.2 Nurse-to-room assignment

Each room $r \in R$ is assigned to exactly one nurse $n \in N(s)$ in each shift $s \in S$ (**H3**), i.e.,

$$\sum_{n \in N(s)} x_{nrs} = 1, \quad \forall r \in R,\, s \in S. \tag{3.9}$$

3.2.3 Nurse-to-patient assignment

A nurse $n \in N$ is assigned to patient $p \in P$ if they are both assigned to the same room $r \in R$ during the same shift $s \in S(n, p)$, i.e.,

$$z_{nps} \geq x_{nrs} + y_{pr} - 1, \quad \forall n \in N,\, p \in P,\, r \in R,\, s \in S(n,p). \tag{3.10}$$

Each patient $p \in P$ is assigned to exactly one nurse $n \in N(s)$ during each shift $s \in S(p)$, i.e.,

$$\sum_{n \in N(s)} z_{nps} = 1, \quad \forall p \in P,\, s \in S(p). \tag{3.11}$$

3.2.4 Continuity of care

The variable a_{np} is set to one if a nurse $n \in N$ is assigned to patient $p \in P$ during at least one shift $s \in S(n,p)$, i.e.,

$$a_{np} \geq z_{nps}, \qquad \forall n \in N,\, p \in P,\, s \in S(n,p), \tag{3.12}$$

$$a_{np} \leq \sum_{s \in S(n,p)} z_{nps}, \quad \forall n \in N,\, p \in P. \tag{3.13}$$

This last constraint is redundant. Since we are minimizing a_{np}, the optimal solution would naturally satisfy this constraint. However, this constraint was kept in the formulation as it was also used in the formulation by Brandt, Klein, et al. (2025). Furthermore, keeping this constraint was sometimes faster. For example, if instance test01 was run for 1 minute, the formulation that included this constraint obtained a gap of 8.6%, while the formulation without it obtained a gap of 9.7%.

3.2.5 Gender-mixing

The variables f_{rs} and m_{rs} are set correctly based on the genders of the patients assigned to room $r \in R$ during early shift $s \in S_{\text{early}}$, i.e.,

$$y_{pr} \leq f_{rs}, \quad \forall p \in P_f,\, r \in R,\, s \in S_{\text{early}}(p). \tag{3.14}$$

$$y_{pr} \leq m_{rs}, \quad \forall p \in P_m,\, r \in R,\, s \in S_{\text{early}}(p), \tag{3.15}$$

The variable v^{gm}_{rs} is set correctly based on the genders of the assigned patient to room $r \in R$ during early shift $s \in S_{\text{early}}$, i.e.,

$$f_{rs} + m_{rs} \leq 1 + v^{\text{gm}}_{rs}, \quad \forall r \in R,\, s \in S_{\text{early}}. \tag{3.16}$$

Because we want to minimize the value v^{gm}_{rs}, the variables f_{rs}, m_{rs} and v^{gm}_{rs} are naturally set to the correct values.

3.2.6 Skill violations

The variable v^{sv}_{ps} for patient $p \in P$ during shift $s \in S(p)$ is set correctly based on the skill level of the assigned nurse $n \in N(s)$, i.e.,

$$v^{\text{sv}}_{ps} \geq \sigma_{\text{req}}(p,s) - \sum_{n \in N(s)} \sigma(n) \cdot z_{nps}, \quad \forall p \in P,\, s \in S(p). \tag{3.17}$$

Because we want to minimize the value v^{sv}_{ps}, this variable is naturally set to the skill violation for a patient $p \in P$ during shift $s \in S(p)$.

3.2.7 Workload violations

The variable v^{wv}_{ns} is set correctly based on how much the workload is exceeded for nurse $n \in N$ during shift $s \in S(n)$, i.e.,

$$\sum_{p \in P(s)} w(p,s) \cdot z_{nps} \leq w_{\max}(n,s) + v^{\text{wv}}_{ns}, \quad \forall n \in N,\, s \in S(n). \tag{3.18}$$

Because we want to minimize the value v^{wv}_{ns}, this variable is naturally set to the workload violation for a nurse $n \in N$ during shift $s \in S(n)$.

3.2.8 Workload imbalance

The variables l_s^- and l_s^+ are set correctly based on the relative workload of the nurses working during shift $s \in S$, i.e.,

$$l_s^- \leq \sum_{p \in P(s)} \frac{w(p,s)}{w_{\max}(n,s)} \cdot z_{nps}, \quad \forall n \in N,\, s \in S(n), \tag{3.19}$$

$$l_s^+ \geq \sum_{p \in P(s)} \frac{w(p,s)}{w_{\max}(n,s)} \cdot z_{nps}, \quad \forall n \in N,\, s \in S(n). \tag{3.20}$$

Because we want to minimize the difference between l_s^- and l_s^+, the variables are naturally set to the smallest and largest relative workload in shift $s \in S$, respectively.

3.2.9 Objective components

The goal is to minimize the continuity of care, gender-mixing, skill violations, workload violations and workload imbalance objective components. These are denoted with g_{cc}, g_{gm}, g_{sv}, g_{wv} and g_{wi}, respectively. These can be computed in the following way:

$$g_{\text{cc}} = \sum_{n \in N} \sum_{p \in P} a_{np} \tag{3.21}$$

$$g_{\text{gm}} = \sum_{r \in R} \sum_{s \in S_{\text{early}}} v_{rs}^{\text{gm}} \tag{3.22}$$

$$g_{\text{sv}} = \sum_{p \in P} \sum_{s \in S(p)} v_{ps}^{\text{sv}} \tag{3.23}$$

$$g_{\text{wv}} = \sum_{n \in N} \sum_{s \in S(n)} v_{ns}^{\text{wv}} \tag{3.24}$$

$$g_{\text{wi}} = \sum_{s \in S} l_s^+ - l_s^- \tag{3.25}$$

Each of these components can be differently weighted with weights $\beta_{\text{cc}}, \beta_{\text{gm}}, \beta_{\text{sv}}, \beta_{\text{wv}}$ and β_{wi}, respectively. The goal is then to minimize the weighted sum, i.e., minimize

$$\beta_{\text{cc}} g_{\text{cc}} + \beta_{\text{gm}} g_{\text{gm}} + \beta_{\text{sv}} g_{\text{sv}} + \beta_{\text{wv}} g_{\text{wv}} + \beta_{\text{wi}} g_{\text{wi}}. \tag{3.26}$$

The full ILP formulation is summarized in Formulation (3.27).

$$\begin{aligned}
\min \quad & \beta_{\text{cc}} g_{\text{cc}} + \beta_{\text{gm}} g_{\text{gm}} + \beta_{\text{sv}} g_{\text{sv}} + \beta_{\text{wv}} g_{\text{wv}} + \beta_{\text{wi}} g_{\text{wi}} \\
\text{s.t.} \quad & \text{Constraints (3.5)-(3.25) hold,} \\
& x_{nrs} \in \{0,1\}, && \forall n \in N,\, r \in R,\, s \in S(n), \\
& y_{pr} \in \{0,1\}, && \forall p \in P,\, r \in R, \\
& z_{nps} \in \{0,1\}, && \forall n \in N,\, p \in P,\, s \in S(n,p), \\
& f_{rs},\, m_{rs} \in \{0,1\}, && \forall\, r \in R,\, s \in S_{\text{early}}, \\
& a_{np} \in \{0,1\}, && \forall\, n \in N,\, p \in P, \\
& v_{rs}^{\text{gm}} \in \{0,1\}, && \forall\, r \in R,\, s \in S_{\text{early}}, \\
& v_{ps}^{\text{sv}} \geq 0, && \forall\, p \in P,\, s \in S(p), \\
& v_{ns}^{\text{wv}} \geq 0, && \forall\, n \in N,\, s \in S(n), \\
& l_s^-,\, l_s^+ \geq 0, && \forall\, s \in S.
\end{aligned} \tag{3.27}$$

In Table 3.2, the size of the ILP formulation is shown for the ten test instances. It shows the number of variables and constraints in the model, as well as the time needed to create the model in Gurobi (Gurobi Optimization, LLC, 2026). Note that, for the implementation, variables g_{cc}, g_{gm}, g_{sv}, g_{wv} and g_{wi} were omitted. Instead, the equations from Constraints (3.21)-(3.25) are directly used in the objective. For this reason, these variables and constraints are not included in Table 3.2.

Table 3.2: Number of variables and constraints, and the model creation time for each test instance.

Instance	Variables		Constraints	Creation time
	Binary	Continuous		
test01	3,280	797	11,637	0.20s
test02	3,900	738	16,285	0.34s
test03	2,408	471	7,911	0.13s
test04	6,111	903	29,984	0.57s
test05	2,902	583	10,817	0.19s
test06	8,816	1,099	45,589	0.71s
test07	10,702	1,478	58,056	1.09s
test08	12,017	1,591	65,986	1.17s
test09	24,647	2,253	196,226	6.62s
test10	252,189	7,129	5,832,233	120.22s

Chapter 4

Complexity, bounds and computational analysis

In this chapter, the formulation from Chapter 3 is used to analyze the problem. To show the difficulty of the problem, Section 4.1 provides several proofs that show that the problem is NP-hard. In Section 4.2, lower and upper bounds are computed using two different methods. Lastly, in Section 4.3, the ILP from the previous chapter is solved.

4.1 Complexity

Previous research has already shown that the PRA problem is NP-hard (Vancroonenburg et al., 2014). Additionally, common NPA formulations are a version of the generalized assignment problem which is known to be NP-hard (Fisher et al., 1986). However, since we use slightly different constraints and objective components, we provide several proofs for the NP-hardness of the PNRA problem. Each proof considers a different objective component and proves that minimizing only that component is already NP-hard. This indicates how difficult it is to optimize the weighted objective function. The first proof shows that optimizing only the continuity of care objective is NP-hard, i.e., when all other weights are set to zero, the problem is still NP-hard.

Lemma 4.1.1. *The PNRA problem is NP-hard, even if only the continuity of care is minimized and there is only a single patient and room.*

Proof. Since there is only one patient p and one room r, the room assignment is clear. The only decision that has to be made is which nurse must care for the patient during each shift. Let $S(p)$ be the set of shifts of patient p's admission and let $S(n,p)$ be the set of shifts that nurse n is scheduled to work during patient p's admission. The goal is to find a subset of nurses $\mathcal{N}(p) \subseteq N$ such that each shift of patient p's stay is covered, i.e., $\bigcup_{n \in \mathcal{N}(p)} S(n,p) = S(p)$. Minimizing the size of this set $\mathcal{N}(p)$ is equivalent to the set cover problem, which is known to be NP-hard (Karp, 1972). This problem is therefore also NP-hard. □

Although the problem of minimizing only the continuity of care objective is NP-hard, this is not true for all objective components. For example, minimizing only the number of skill violations can very easily be done by assigning the most qualified

nurse present during every shift to all patients. However, minimizing the other objective components is NP-hard. This is proven in Lemmas 4.1.2, 4.1.3 and 4.1.4.

Lemma 4.1.2. *The PNRA problem is NP-hard, even if only the workload imbalance is minimized, each nurse has the same maximum workload and the patient-to-room assignment is fixed.*

Proof. Since the workload imbalance can be minimized in each shift separately when the patient-to-room assignment is fixed, we show that the problem is NP-hard for each shift $s \in S$. Let $w(r, s)$ be the total workload of all patients assigned to room $r \in R$ during shift s based on the fixed patient-to-room assignment. A nurse-to-room assignment is equivalent to partitioning the set of rooms R into the sets $R_n(s)$ for each nurse $n \in N(s)$. The set $R_n(s)$ then denotes which rooms nurse n is assigned to during shift s. The assigned workload of nurse n is then

$$\sum_{r \in R_n(s)} w(r, s). \tag{4.1}$$

Minimizing the workload imbalance is then equivalent to finding a partition such that

$$\max_{n \in N(s)} \sum_{r \in R_n(s)} w(r, s) - \min_{n \in N(s)} \sum_{r \in R_n(s)} w(r, s) \tag{4.2}$$

is minimized. This is equivalent to a multiway number partitioning problem, which is known to be NP-hard (Karp, 1972). Note that for minimizing the workload imbalance we normally divide the assigned workload by the maximum workload of the nurse. However, since the maximum workload of each nurse is assumed to be the same, this is not needed. □

Lemma 4.1.3. *The PNRA problem is NP-hard, even if only the number of workload violations is minimized and the patient-to-room assignment is fixed.*

Proof. Since the number of workload violations can be minimized in each shift separately, we show that the problem is NP-hard for each shift s. Recall that an optimization problem is NP-hard if the associated decision problem is NP-complete. Therefore, we show that deciding whether a solution exists with no workload violation is NP-complete. Let $w(r, s)$ be defined as in the proof of Lemma 4.1.2. Finding a nurse-to-room assignment with no workload violation is equivalent to partitioning the set of rooms R into the sets $R_n(s)$ for each nurse $n \in N(s)$, such that the workload does not exceed the maximum load, i.e.,

$$\sum_{r \in R_n(s)} w(r, s) \leq w_{\max}(n, s). \tag{4.3}$$

This is equivalent to the bin packing problem with varying bin sizes, where the nurses are the bins, the rooms are the items, the workload in a room is the size of the item and the maximum workload of a nurse is the bin capacity. The bin packing problem is known to be NP-complete (Garey & Johnson, 1979). This shows that the problem is indeed NP-hard. □

Lemma 4.1.4. *The PNRA problem is NP-hard, even if only the number of mixed-gender rooms is minimized.*

Proof. Since we are only minimizing the number of mixed-gender rooms, we can ignore the nurse-to-room assignment. Similarly to Lemma 4.1.3, we prove that the corresponding decision problem is NP-complete. Therefore, we show that it is NP-complete to decide whether a patient-to-room assignment with no gender-mixed rooms exists. We must therefore show that this decision problem is both in NP and NP-hard.

Unlike the other NP-hardness proofs, we provide a more traditional proof by giving a reduction from the partition problem, which is known to be NP-hard (Karp, 1972). An instance of the partition problem consists of n integers $a_1, \ldots, a_n$ and an integer b such that $2b = \sum_{i=1}^{n} a_i$. The question is whether we can find two sets A and B such that $\sum_{i \in A} a_i = \sum_{i \in B} a_i = b$. We reduce such an instance to a decision version of the PNRA problem consisting of only a single day, where we have

- n rooms with capacity a_i for $i = 1, \ldots, n$;
- b male patients and b female patients. Each patient has no incompatible rooms.

The question is whether there exists a patient-to-room assignment such that there is no gender-mixed room. This reduction is clearly polynomial. Now we prove that an instance is a yes-instance of the partition problem if and only if the reduced instance is a yes-instance to the PNRA problem.

$\implies$ Suppose there exists a partition A and B such that $\sum_{i \in A} a_i = \sum_{i \in B} a_i = b$. We create a solution to the PNRA problem by assigning all male patients arbitrarily to a room $i \in A$ until the rooms are all filled. Because $\sum_{i \in A} a_i = b$ there is exactly enough capacity for each patient. The same can be done for the female patients by placing them in the rooms from B. This yields a solution with no gender-mixed rooms.

$\impliedby$ Suppose we find a solution to the PNRA problem where no rooms have patients with multiple genders. Let A be the set of rooms containing all male patients and B be the set containing all female patients. Because the capacity of the rooms must cover the patients, we must have $\sum_{i \in A} a_i \geq b$ and $\sum_{i \in B} a_i \geq b$. However, because the total capacity is $2b$, we must have that $\sum_{i \in A} a_i = b$ and $\sum_{i \in B} a_i = b$, which yields a feasible solution to the partition problem.

Because we have a polynomial-time reduction from the partition problem, which is NP-hard, the decision problem is also NP-hard. Furthermore, because a yes-instance can be verified in polynomial time, it is also NP-complete. This shows that minimizing the number of gender-mixed rooms is NP-hard. □

We see that the PNRA problem is NP-hard, even when only a single objective component is minimized, except for the skill violations objective. This highlights how difficult it is to optimize them all simultaneously.

4.2 Objective bounds

We can compute lower and upper bounds for each individual objective function component. We determine bounds for the continuity of care (g_{cc}), the gender-mixing (g_{gm}), the skill violations (g_{sv}), the workload violations (g_{wv}) and workload imbalance (g_{wi}). This allows us to find a range in which each of these objective components lie, which can help a planner to determine suitable weights for each objective component. Furthermore, we provide a lower bound on the combined objective function. For this, we assume that the weights are all set to 1. The impact of different weights is analyzed in Section 6.2.

For the computation of these lower and upper bounds, several ILPs are created. As they are often derived directly from Formulation (3.27), they are often not given here. They can be found in Appendix A.

4.2.1 Lower bounds

Two different methods for computing a lower bound on the objective function components are considered. The first method is a linear programming (LP) relaxation, which relaxes the integrality of the variables in the ILP. The second method relaxed or removes several constraints instead of relaxing integrality. Because we consider a subset of the original problem, we refer to this as a partial relaxation.

Firstly, a lower bound for each objective component is computed. This is done by setting all other weights to zero. Later, the lower bound is computed for the combined objective function, where each weight is set to 1. In Table 4.1 the lower bounds for each objective component using the LP relaxation can be seen for each of the ten test instances.

Table 4.1: Lower bounds on the objective components, obtained using the LP relaxation of Formulation (3.27).

Instance	Continuity of care	Gender-mixing	Skill violations	Workload violations	Workload Imbalance
test01	159.5	0	2	0	0
test02	154.25	0	33	0	0
test03	109	0	2	0	0
test04	210.156	0	0	0	0
test05	131.233	0	3	0	0
test06	336	0	0	0	0
test07	321	0	177	0	0
test08	357.045	0	0	0	0
test09	426.732	0	0	0	0
test10	1565.147	0	0	0	0

We see that for most of the objective components, we find a lower bound of zero. This might suggest that the LP relaxation has too much freedom. The fact that the continuity of care objective has a nonzero lowerbound is not surprising. This namely follows from Constraints (3.11) and (3.12). The first shows that for each patient $p \in P$ and shift $s \in S(p)$, there is a nurse with $z_{nps} > 0$. Combining this with the second constraint implies that for each patient $p \in P$, there must be a nurse $n \in N$ with $a_{np} > 0$. For some instances, the lower bound on g_{sv} is also nonzero. This is because, for those instances, there are shifts in which there are no nurses with a high enough skill level to meet the patients' requirements.

Almost all lower bounds could be computed within a few seconds. However, this took much more time for instance test10. For the gender, skill, workload and imbalance objectives, it took somewhere between two and three minutes to compute the lower bounds. The continuity of care objective however, took almost four hours to complete. This is due to the large number of variables and constraints (see Table 3.2), which makes solving the problem difficult.

Generally, the LP relaxation both provides poor lower bounds and can take much time to obtain those lower bounds. Instead, the lower bounds can be computed using a partial relaxation, where several constraints of the problem are relaxed or removed while integrality is kept. To start, we can use the idea behind the proof of Lemma 4.1.1 to compute a lower bound on the continuity of care objective value. By determining the smallest set cover $\mathcal{N}(p)$ for each patient, we obtain a lower bound for g_{cc}. This is formulated as an ILP in Formulation (4.4). This formulation yields a lower bound, since we ignore the patient-to-room assignment and relax the assumption that all patients in the same room are cared for by the same nurse. In fact, the exact nurse that is assigned to a patient in a shift is not important. It only matters that there is at least one nurse in the set cover $\mathcal{N}(p)$ that can be assigned to patient p in every shift.

$$\begin{aligned} \min \quad & \sum_{n \in N} \sum_{p \in P} a_{np} \\ \text{s.t.} \quad & \sum_{n \in N(s)} a_{np} \geq 1, \qquad \forall p \in P,\, s \in S(p), \\ & a_{np} \in \{0,1\}, \quad \forall n \in N,\, p \in P. \end{aligned} \tag{4.4}$$

Note that, although this is still a set cover problem (and therefore NP-hard), this problem is much easier to solve compared to the full PNRA problem. This is mainly because both the patient-to-room assignment and nurse-to-room assignments are ignored. Although this is not a direct subset of the constraints from the ILP in Chapter 3, it can be obtained by summing Constraint (3.12) over $N(s)$ and using Constraint (3.11). As such, it is still considered a partial relaxation.

To obtain lower bounds on the total workload violation (g_{wv}) and workload imbalance (g_{wi}), we can use a similar partial relaxation by directly assigning nurses to patients. This allows us to ignore the patient-to-room assignment and any associated variables and constraints. In particular, we only need to keep the z_{nps} variables together with Constraint (3.11). Furthermore, depending on whether we want to find a lower bound for the workload violation or the workload imbalance, we must add variables v_{ns}^{wv} or variables l_s^- and l_s^+, together with Constraint (3.18) or Constraints (3.19) and (3.20). The ILP formulations for the workload violation and imbalance lower bounds are written out in Formulations (A.1) and (A.2), respectively.

Note that, since we are assigning nurses to patients directly and the patient-to-room assignment is ignored, each shift could be minimized independently. For calculating the workload imbalance lower bound in Table 4.2, this was done to reduce the computation time. For the workload violation lower bound, this was not needed since the lower bound could be computed very quickly.

Computing a lower bound for the total skill violation (g_{sv}) is much easier. As was stated in Section 4.1, this objective can be minimized by assigning each patient to the most skilled nurse present during each shift.

Lastly, we compute a lower bound on the total gender-mixing violation (g_{gm}). Recall that in the IHTC (Ceschia et al., 2025), the gender-mixing constraint was formulated in a hard way. Because we use the datasets from this competition, we know that it is possible to find a patient-to-room assignment such that this objective component is zero for our instances. However, for new problem instances we do not know if this is also possible. For these instances, we can create a simple ILP in which only the variables and constraints related to the patient-to-room assignment and gender-mixing are used to compute a lower bound. This ILP is written out in Formulation (A.3).

The lower bounds for each individual objective component using the partial relaxations described above are shown in Table 4.2. Almost all lower bounds are computed within a few seconds. For instance test10, computing the lower bound took roughly 30 seconds for the gender-mixing objective component and roughly 3 minutes for the workload imbalance objective component. However, this is still faster than computing the LP lower bound. This is especially true for computing the continuity of care lower bound, which took almost four hours for the LP relaxation and less than a second for the partial relaxation.

Table 4.2: Lower bounds on the objective components, obtained using partial relaxations.

Instance	Continuity of care	Gender-mixing	Skill violations	Workload violations	Workload imbalance
test01	163	0	2	0	3.633
test02	166	0	33	0	2.767
test03	109	0	2	0	3.283
test04	225	0	0	0	3.183
test05	134	0	3	0	2.550
test06	343	0	0	0	2.967
test07	331	0	177	0	4.617
test08	373	0	0	0	4.200
test09	452	0	0	0	4.650
test10	1618	0	0	0	6.217

For every instance and every objective component, the lower bounds obtained using the partial relaxation are better, or at least as good as, the lower bounds from the LP relaxation. Additionally, the lower bounds from Table 4.2 could be computed faster than those in Table 4.1. This is because the problem size for the ILPs used in the partial relaxation is much smaller, since Constraint (3.10) is removed, which in the case of instance test10 removes 5,559,008 constraints.

Note that, although for these test instances, the lower bound obtained using the partial relaxation is always tighter than the LP relaxation, this might not always be the case. To show this, we provide a counterexample for the continuity of care lower bound.

Example 4.2.1. *Consider a problem instance consisting of a single room r, two nurses n_1 and n_2, and two patients p_1 and p_2. The days for which these nurses and patients are scheduled is shown in Table 4.3. For simplicity, we assume that each day consists of only a single shift. These shifts are denotes with s_1, s_2 and s_3.*

Table 4.3: An example nurse and patient schedule.

Day	1	2	3
n_1	✓	✓	
n_2		✓	✓

Day	1	2	3
p_1	✓	✓	
p_2		✓	✓

First, we calculate the minimum set cover for each patient. Observe that, for patient p_1, nurse n_1 is present on both days of their stay. This gives minimum set cover $\mathcal{N}(p_1) = \{n_1\}$. Similarly, we can obtain $\mathcal{N}(p_2) = \{n_2\}$. This gives a lower bound of 2 on the continuity of care objective component when using the partial relaxation.

Next, we show that the LP relaxation obtains a higher bound. To do this, we manipulate various constraints. Since there is only a single nurse on days 1 and 3, we must have that

$$z_{n_1 p_1 s_1} = 1, \tag{4.5}$$
$$z_{n_2 p_2 s_3} = 1, \tag{4.6}$$

because of Constraint (3.11)*. This, together with Constraint* (3.12) *implies that*

$$a_{n_1 p_1} = 1, \tag{4.7}$$
$$a_{n_2 p_2} = 1. \tag{4.8}$$

To calculate the other values of a_{np}, first note that, since there is only a single room r,

$$\begin{aligned} z_{nps} &\geq x_{nrs} + y_{pr} - 1, \quad \forall n \in N, p \in P, s \in S(n,p), \\ &\geq x_{nrs}. \end{aligned} \tag{4.9}$$

When summing over $N(s)$ and using Constraints (3.9) *and* (3.11)*, we have that*

$$1 = \sum_{n \in N(s)} z_{nps} \geq \sum_{n \in N(s)} x_{nrs} = 1, \quad \forall p \in P,\, s \in S(p), \tag{4.10}$$

which implies

$$z_{nps} = x_{nrs}, \quad \forall n \in N, p \in P, s \in S(n,p). \tag{4.11}$$

Therefore, we also have that

$$a_{np} \geq x_{nrs}, \quad \forall n \in N, p \in P, s \in S(n,p). \tag{4.12}$$

Since we are considering a minimization problem, this means that

$$a_{np} = \max_{s \in S(n,p)} x_{nrs}, \quad \forall n \in N, p \in P. \tag{4.13}$$

Since $S(n_1, p_2) = S(n_2, p_1) = \{s_2\}$, we therefore have the following equations

$$a_{n_2 p_1} = x_{n_2 r s_2}, \tag{4.14}$$
$$a_{n_1 p_2} = x_{n_1 r s_2}. \tag{4.15}$$

Combining this with the other values from Equations (4.7) *and* (4.8)*, gives that the continuity of care value is*

$$\sum_{n \in N} \sum_{p \in P} a_{np} = 2 + x_{n_1 r s_2} + x_{n_2 r s_2}. \tag{4.16}$$

However, since Constraint (3.9) *states that* $x_{n_1 r 2} + x_{n_2 r 2} = 1$*, the objective value becomes* 3*. This means that the lower bound from the LP relaxation for the continuity of care objective is* 3*. This is a higher lower bound than the one obtained using the partial relaxation.*
The reason why this happens in this instance is because the z_{nps} *variables in the LP relaxation are much more constrained by* x_{nrs} *than for the test instances. In those test instances, the value for* y_{pr} *could often be set to a small value (while ensuring that* $\sum_{r \in R} y_{pr} = 1$*). This means that Constraint* (3.10) *is not very strict. Variables* z_{nps} *could then be set to the desired nurse, while mostly ignoring the value for* x_{nrs}*.*

Although the partial relaxation achieves a better bound for each individual objective component, this might not be the case when considering the full objective function, where all weights are set to 1. The LP relaxation can be simply computed by solving Formulation (3.27) where each binary variable is instead continuous. Defining the partial relaxation of the full objective function requires more work. Similar to what was done previously, the partial relaxation ignores the patient-to-room and nurse-to-room assignment, and instead assigns patients to nurses directly. This means that any constraint that uses variables y_{pr} or x_{nrs} is omitted. The resulting ILP can be found in Formulation (A.4).

Note that, since computing the gender-mixing objective requires the patient-to-room assignment, the partial relaxation cannot take this objective into account. For this reason, the lower bound from Table 4.2 is added to the result separately. Although this lower bound is always zero for these instances, this might not be the case for other instances.

In Table 4.4, the obtained lower bounds on the full objective function using both the LP and partial relaxation are given. The partial relaxation could not be solved within a reasonable time, even for smaller instances. For this reason, the solving time is limited to two minutes and the best bound obtained at that time is reported. Note that the runtime reported in Table 4.4 can exceed two minutes since it also includes the time to create the ILP model and the time to compute the gender-mixing lower bound.

Table 4.4: Lower bounds on combined objective value, obtained using LP relaxation and partial relaxation.

Instance	LP relaxation		Partial relaxation	
	Lower bound	Runtime (s)	Lower bound	Runtime (s)
test01	203.314	0.555	223.100	6.22
test02	285.181	0.662	304.050	87.773
test03	133.347	0.246	137.333	1.049
test04	291.930	1.229	308.717	120.225
test05	160.636	0.489	164.983	5.149
test06	369.611	2.025	375.017	121.755
test07	684.461	2.563	724.000	120.345
test08	479.974	3.235	482.283	120.469
test09	559.775	12.600	600.267	120.916
test10	1889.651	9246.595	1967.250	152.697

We see that for all instances, the partial relaxation again produces tighter lower bounds, although often more time was needed to obtain them. The only exception to this is instance test10. Due to the large number of variables and constraints, computing the relaxed objective value takes over 2.5 hours. Since ILP solvers rely on quickly solving the LP relaxation, this is a big problem in Section 4.3, where ILP Formulation (3.27) is solved.

4.2.2 Upper bounds

Upper bounds can be used, together with the lower bounds, to determine the range in which each objective component lies. This could help planners determine suitable weights to give to each objective component. Note that this section does not attempt to provide upper bounds on the optimal objective value, for example by providing feasible solutions, but instead provides worst case scenarios for each objective component. The same is true for the full objective function.

For the skill violations, workload violations and workload imbalance objective components, computing an upper bound is easy and can be done in the following ways:

- Skill violations: In each shift, assign all patients to the nurse with the lowest skill level present.
- Workload violations: In each shift, assign all patients to the nurse with the lowest maximum workload present.
- Workload imbalance: In each shift, assign all patients to the nurse with the lowest maximum workload present.

Note that for computing these upper bounds, the room assignment does not matter since we are assigning all patients to the same nurse anyway. This is therefore a valid solution that gives a tight bound on the worst-case objective component value.

For the continuity of care and gender-mixing objective components, computing an upper bound is not as easy. Currently, Formulation (4.4) is not suitable for maximization. With the current constraints, each variable a_{np} could be set to 1, which would give $|P| \cdot |N|$ as an upper bound. This could be improved by reintroducing the

nurse-to-patient assignment variable z_{nps} together with Constraints (3.11), (3.12) and (3.13). This ensures that only a single nurse can be assigned to a patient in each shift. The resulting ILP is given in Formulation (A.5).

Note that, similar to the lower bound calculation, this is a partial relaxation since the nurses are directly assigned to patients. This means that there are instances for which the resulting upper bound is not actually attainable. This is the case for the instance from Example 4.2.1. In this example, the partial upper bound gives a value of 4, namely two distinct nurses for each patient. However, due to the shared nurse on day 2, the largest attainable continuity of care objective value is only 3.

To provide a suitable lower bound on the gender-mixing objective component, we must also use an ILP. For this, we need the patient-to-room assignment variables y_{pr} and the variables related to the gender-mixing objective, namely f_{rs}, m_{rs} and v_{rs}^{gm}. The related constraints are Constraints (3.5)-(3.8) and Constraints (3.14)-(3.16). This has the same problem as the continuity of care upper bound, namely that variables f_{rs}, m_{rs} and v_{rs}^{gm} could always be set to 1. To fix this, the following constraints are added:

$$\sum_{p \in P_f(s)} y_{pr} \geq f_{rs}, \quad \forall r \in R,\, s \in S_{\text{early}}, \tag{4.17}$$

$$\sum_{p \in P_m(s)} y_{pr} \geq m_{rs}, \quad \forall r \in R,\, s \in S_{\text{early}}, \tag{4.18}$$

$$v_{rs}^{\text{gm}} \leq f_{rs}, \quad \forall r \in R,\, s \in S_{\text{early}}, \tag{4.19}$$

$$v_{rs}^{\text{gm}} \leq m_{rs}, \quad \forall r \in R,\, s \in S_{\text{early}}. \tag{4.20}$$

Adding these constraints ensures that each variable is set correctly. In particular, the first two constraints state that variables f_{rs} and m_{rs} can only be set to 1 if at least one patient of the corresponding gender is assigned to room $r \in R$ during early shift $s \in S_{\text{early}}$. The other two constraints ensure that v_{rs}^{gm} can only be set to 1 if both $f_{rs} = 1$ and $m_{rs} = 1$. Constraints (3.14)-(3.16) are removed as they only provide lower bounds on f_{rs}, m_{rs} and v_{rs}^{gm}, which are naturally satisfied when maximizing. The resulting ILP is given in Formulation (A.6). Unlike the continuity of care upper bound, this upper bound is attainable because is it constructed using a valid patient-to-room assignment.

The partial upper bounds for each individual objective component are given in Table 4.5.

Table 4.5: Upper bounds on the objective components, obtained using partial relaxations.

Instance	Continuity of care	Gender-mixing	Skill violations	Workload violations	Workload imbalance
test01	410	59	234	374	117.300
test02	469	66	363	587	112.250
test03	247	20	95	165	53.167
test04	618	82	481	928	177.017
test05	338	37	125	284	79.000
test06	817	102	254	1166	157.100
test07	1023	138	1569	1549	308.717
test08	1117	130	394	1624	225.400
test09	1658	223	1854	2846	561.817
test10	5655	791	9720	10642	2157.900

In addition to the upper bound for each individual component, we also compute an upper bound on the full objective function as a whole. We again assume that all weights are set to 1. The upper bound is used in Chapter 5, together with the lower bound, to normalize the objective value such that it can be compared with other instances.

When the ILP Formulation (3.27) is used directly with maximization instead of minimization, almost all of the penalty variables could be set to 1. To prevent this, additional constraints must be added. For the gender-mixing objective, Constraints (4.17)-(4.20) could be added. For the skill violations, we could write Constraint (3.17) as

$$v^{\text{sv}}_{ps} = \sum_{n \in N(s)} \max\{0,\ \sigma_{\text{req}}(p,s) - \sigma(n)\} \cdot z_{nps}, \quad \forall p \in P, s \in S(p). \tag{4.21}$$

The workload violation and imbalance objectives are slightly harder to fix. To do this, we need to introduce additional binary variables δ^{wv}_{ns} for $n \in N$ and $s \in S(n)$. These variables indicate whether the assigned workload of nurse $n \in N$ exceeds their maximum workload in a shift $s \in S(n)$. Furthermore, we define constants

$$M^{\text{wv}}_{ns} := \max\left\{ \Big| \sum_{p \in P(s)} w(p,s) - w_{\max}(n,s) \Big|,\ w_{\max}(n,s) \right\}, \tag{4.22}$$

for $n \in N$ and $s \in S(n)$. This allows us to add the constraints

$$v^{\text{wv}}_{ns} \leq M^{\text{wv}}_{ns} \cdot \delta^{\text{wv}}_{ns}, \tag{4.23}$$

$$v^{\text{wv}}_{ns} \leq \sum_{p \in P(s)} (w(p,s) \cdot z_{nps}) - w_{\max}(n,s) + M^{\text{wv}}_{ns} \cdot (1 - \delta^{\text{wv}}_{ns}), \tag{4.24}$$

for $n \in N$ and $s \in S(n)$. This ensures that v^{wv}_{ns} is set to the correct value. If $\delta^{\text{wv}}_{ns} = 0$ for nurse $n \in N$ and shift $s \in S(n)$, i.e., if their maximum workload is not exceeded, Constraint (4.23) implies that $v^{\text{wv}}_{ns} = 0$. Since $M^{\text{wv}}_{ns} \geq w_{\max}(n,s)$, Constraint (4.24)

does not conflict with this. If instead $\delta^{\text{wv}}_{ns} = 1$ and nurse n's maximum workload is exceeded, then we must have that

$$v^{\text{wv}}_{ns} = \sum_{p \in P(s)} (w(p,s) \cdot z_{nps}) - w_{\max}(n,s). \tag{4.25}$$

Since M^{wv}_{ns} is bigger than the largest possible difference in assigned and maximum workload, Constraint (4.23) does not contradict this.

Apart from these constraints, we also add constraints

$$\sum_{n \in N(s)} v^{\text{wv}}_{ns} \leq \max \left\{ 0, \sum_{p \in P(s)} w(p,s) - \min_{n \in N(s)} w_{\max}(n,s) \right\}, \tag{4.26}$$

for each $s \in S$. For the LP relaxation, this ensures that the total workload violation in a shift does not exceed the upper bound found in Table 4.5.

For the workload imbalance, we similarly add binary variables δ^{-}_{ns} and δ^{+}_{ns} for $s \in S$ and $n \in N(s)$. These indicate which nurse has the lowest or highest relative workload in a shift $s \in S$, respectively. We must therefore add constraints

$$\sum_{n \in N(s)} \delta^{-}_{ns} = 1, \qquad \forall s \in S, \tag{4.27}$$

$$\sum_{n \in N(s)} \delta^{+}_{ns} = 1, \qquad \forall s \in S. \tag{4.28}$$

This ensures that only one nurse is selected in each shift. Additionally, we define constants

$$M^{\text{wi}}_{s} := \frac{\sum_{p \in P(s)} w(p,s)}{\min_{n \in N(s)} w_{\max}(n,s)}, \tag{4.29}$$

for $s \in S$, which calculates the largest possible relative workload in a shift $s \in S$. This allows us to add constraints

$$l^{-}_{s} \geq \sum_{p \in P(s)} \left(\frac{w(p,s)}{w_{\max}(n,s)} \cdot z_{nps} \right) - M^{\text{wi}}_{s}(1 - \delta^{-}_{ns}), \tag{4.30}$$

$$l^{+}_{s} \leq \sum_{p \in P(s)} \left(\frac{w(p,s)}{w_{\max}(n,s)} \cdot z_{nps} \right) + M^{\text{wi}}_{s}(1 - \delta^{+}_{ns}), \tag{4.31}$$

for all $s \in S$ and $n \in N(s)$. If δ^{-}_{ns} (or δ^{+}_{ns}) is set to zero, this constraint adds no new information. But if it is set to one, then l^{-}_{s} (or l^{+}_{s}) is set to that nurse's relative workload.

Adding these variables and constraints to the ILPs for the LP and partial relaxation can produce a upper bound on the objective value. The ILPs can be found in Formulations (A.7) and (A.8). Note that some redundant constraints were removed. Since it is a maximization problem, only upper bounds are needed to get the correct variable assignment. For this reason, Constraints (3.15)-(3.20) were removed.

In Table 4.6, the resulting upper bounds can be found. The partial relaxation could often not be solved within a reasonable time. Similar to the lower bound calculation, we set a runtime limit of two minutes. Again, the reported runtime

could exceed this limit because it does not include the time needed to create the model and the computation time to obtain the gender-mixing upper bound.

The same conclusions can be drawn as for the full lower bound, namely that the partial relaxation obtains better upper bounds, but may take longer to do so. Also, the time needed to determine the LP relaxation for instance test10 is again very long.

Table 4.6: Upper bounds on combined objective value, obtained using LP relaxation and partial relaxation.

Instance	LP relaxation		Partial relaxation	
	Upper bound	Runtime (s)	Upper bound	Runtime (s)
test01	1160.442	0.496	1072.100	0.550
test02	1572.329	0.617	1452.217	9.184
test03	556.195	0.190	519.000	0.348
test04	2199.737	1.154	2003.700	48.864
test05	847.785	0.419	778.967	0.968
test06	2484.911	1.856	2257.283	120.430
test07	4335.648	2.502	3968.217	120.467
test08	3486.468	4.979	2999.917	120.731
test09	6847.287	12.800	5904.717	123.680
test10	28727.719	6357.503	26102.800	180.649

4.3 Solving the ILP

We solve the ILP from Formulation (3.27) for each of the ten test instances. Since the problem takes very long to solve, the runtime was limited to 8 hours. In Table 4.7, the results are shown.

Table 4.7: Solution results for the 10 test instances, with a runtime limit of 8 hours.

Instance	Best obj.	Best bound	Gap (%)	Runtime
test01	239.883	233.118	2.82	8:00:00.00
test02	355.450	312.550	12.07	8:00:00.00
test03	141.767	141.767	0.00	0:06:05.50
test04	396.883	309.383	22.05	8:00:00.00
test05	175.000	173.366	0.93	8:00:00.00
test06	466.533	376.850	19.22	8:00:00.00
test07	844.950	728.950	13.73	8:00:00.00
test08	626.017	488.617	21.95	8:00:00.00
test09	837.400	616.667	26.36	8:00:00.00
test10	11652.750	1974.850	83.05	8:00:00.00

Only one of the instances is able to be solved within the 8 hour time limit. This instance consisted of the lowest number of variables and constraints (see Table 3.2). Furthermore, in this instance there is a day in which there are no patients present. This means that the problem can essentially be decoupled into two independent subproblems, the days before and the days after. Gurobi might be able to leverage this to speed up the solver.

Out of the other instances, test10 is most notable. For this instance, the gap is quite large compared to the other instances. To explain why this is the case, recall the results from Table 4.4, in which the problem was relaxed to obtain a lower bound. In the case of instance test10, computing the relaxation took over 2.5 hours. Since computing the relaxed solution is an important part of ILP solvers, it was not able to improve the solution much.

In cases where solving the relaxation is difficult, Gurobi (Gurobi Optimization, LLC, 2026) recommends using the NoRel heuristic. This is a heuristic built into the Gurobi MIP solver that does not require computing the relaxed solution. It does so by solving many sub-MIPs in parallel. Since solving the relaxation for this instance takes over 2.5 hours, using this heuristic is especially useful. We ran this heuristic for 8 hours for instance test10. In the end, it found a solution with an objective value of 3442.183. Although the NoRel heuristic also provides a lower bound on the objective value, this is much worse compared to the lower bound from the relaxation. When using the lower bound from Table 4.7, we obtain a gap of 42.63%.

Although this NoRel heuristic is able to find a relatively good solution to the problem, it is a very general heuristic that does not use the structure of the problem to find a solution. In the next chapter, we try to exploit this structure to improve the solutions from Table 4.7 and try to improve the speed at which these solutions are found.

To get a better idea on what the obtained solutions look like, Table 4.8 shows how the objective function value can be decomposed into each of the objective components. Additionally, Figures 4.1 and 4.2 visualize the patient-to-room assignment by showing the number of patients and the gender in each room.

Table 4.8: Decomposition into the different objective components.

Instance	Objective value	Continuity of care	Gender-mixing	Skill violations	Workload violations	Workload imbalance
test01	239.883	198	0	16	0	25.883
test02	355.450	211	2	117	2	23.450
test03	141.767	123	0	7	0	11.767
test04	396.883	284	16	56	12	28.883
test05	175.000	150	0	9	0	16.000
test06	466.533	413	18	18	0	17.533
test07	844.950	473	10	305	9	47.950
test08	626.017	482	15	88	8	33.017
test09	837.400	689	32	60	7	49.400
test10	11652.750	4983	343	3012	3103	211.750
test10 (NoRel)	3442.183	2656	250	422	51	63.183

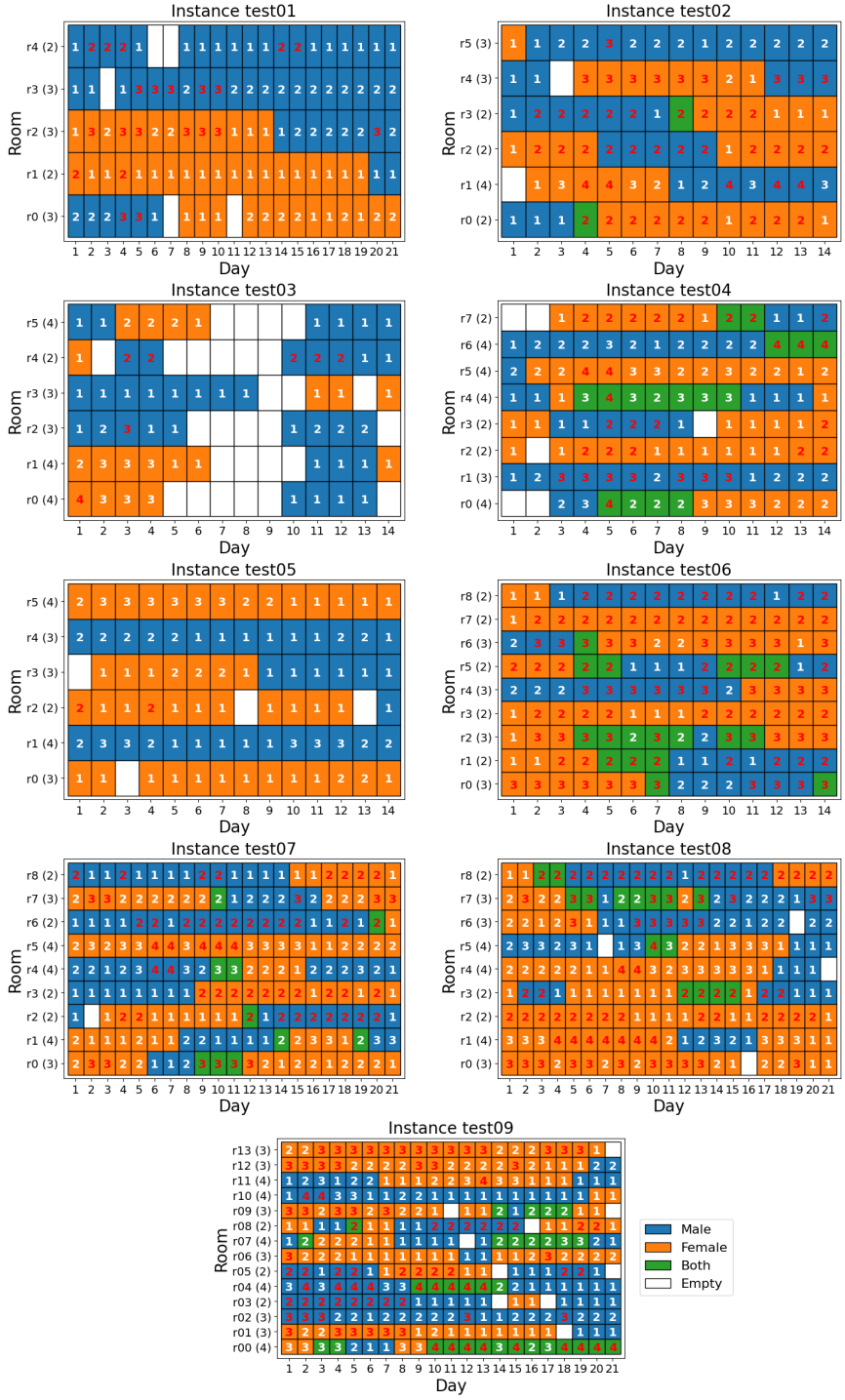


Figure 4.1: Solutions for the test instances from Table 4.7 obtained using ILP Formulation (3.27). It shows the gender (color) and the number of patients (value) in each room on each day. In case the maximum capacity is reached for a room, the value is colored red. The y-axis labels show the room name and its maximum capacity in brackets.

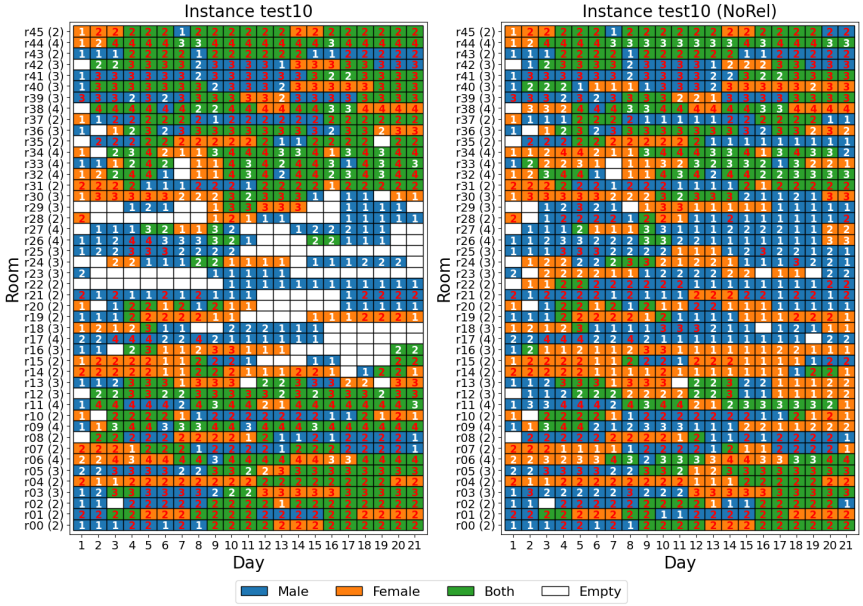


Figure 4.2: Solutions for instance test10 using ILP Formulation (3.27), either with (right) or without (left) the NoRel heuristic. It shows the gender (color) and the number of patients (value) in each room on each day. In case the maximum capacity is reached for a room, the value is colored red. The y-axis labels show the room name and its maximum capacity in brackets.

Chapter 5

Heuristics

In this chapter, several heuristics are developed to solve the PNRA problem. Each heuristic contains several parameters that must be tuned. In Section 5.1, we explain how the data is split into a training and evaluation set, and how results for different instances can be compared. In the next sections, three different heuristics are presented:

- In Section 5.2, a greedy heuristic is used to quickly find a feasible solution. This is done by first determining a feasible patient-to-room assignment and using that to find a good nurse-to-room assignment.
- In Section 5.3, an interval heuristic is introduced. This heuristic splits the planning horizon into several intervals which are solved sequentially.
- In Section 5.4, a simulated annealing algorithm is presented. This heuristic can either be used to find a new solution or improve a known solution.

5.1 Methodology

In each of the heuristics discussed in this chapter, there are several parameters that determine their performance. To determine these parameters, we split our dataset into a training and an evaluation set. The training set is used to tune the parameters of the heuristics and the evaluation set is used to evaluate them. Recall from Section 2.2 that the instances from the IHTC were split into three categories: test, public and hidden instances. Similar to how the IHTC was evaluated, we use the test and public instances for training and use the hidden instances for evaluation.

Because the various instances can be very different in size and complexity, they are additionally divided into three sizes: small, medium and large. Because all other size parameters (number of days, rooms and nurses) are heavily correlated with the number of patients, we use this to split the instances. If an instance has less than 75 patients, it is considered small. If an instance contains at least 75, but less than 240 patients, it is considered medium. All other instances are considered large. These thresholds were determined such that both the training and evaluation datasets contain roughly the same ratio of each size, while also taking into account the runtime for various heuristics. The distribution of instances across each size class can be seen in Figure 5.1. Out of the ten test instances, instances test01-test05 are small, instances test06-test09 are medium and instance test10 is large.

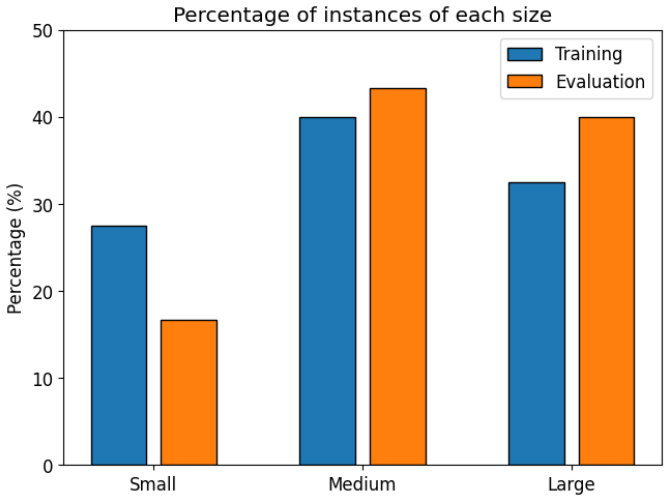


Figure 5.1: Fraction of the training and evaluation set in each size class.

Although the data has now been split into small, medium and large instances, the objective value for each instance within a size class can still be very different. Consider for example instance test03 and test04, which are both small instances. Instance test03 has a best known objective value of 141.767, while test04 has a value of 396.883. If we would evaluate a heuristic by taking the average objective over all instances in the size class, this would give more weight to improvements for instance test04. To account for this, we rescale the objectives. Let l and u be the partial lower and upper bound, respectively. These values can be found in Tables 4.4 and 4.6 for the test instances and in Table B.1 for the public and hidden instances. The objective value x is rescaled by taking

$$\frac{x-l}{u-l}. \tag{5.1}$$

This ensures that this is between 0 and 1. We call this value the objective score. A heuristic can then be evaluated by calculating the average objective score across all instances in a size class. We consider methods or parameters that have a lower average objective score to be better.

5.2 Greedy heuristic

A feasible solution to the PRNA problem consists of two components: the patient-to-room assignment and the nurse-to-room assignment. To construct a solution greedily, we first derive a patient-to-room assignment and then use this assignment to create a nurse-to-room assignment.

5.2.1 Patient-to-room heuristic

In the PRNA problem, there are two hard constraints related to the patient-to-room assignment. These are the room capacity constraint (**H1**) and the incompatible room constraint (**H2**). Out of the soft constraints, there is only one constraint directly related to the patient-to-room assignment, namely the gender-mixing constraint (**S2**). The goal of this heuristic is to find a patient-to-room assignment

satisfying the hard constraints that minimizes the number of gender-mixed rooms, while also allowing for easier optimization for the other objective components (**S1**, **S3**, **S4**, **S5**). The algorithm works as follows:

1. Initialize $f(r, d) = 0$ and $m(r, d) = 0$ to keep track of the number of female and the number of male patients that are assigned to room $r \in R$ on day $d \in D$, respectively. These values are updated each time a patient is assigned to a room.

2. Assign the current occupants to the correct room. The values $f(r, d)$ and $m(r, d)$ are updated accordingly depending on the patients' genders.

3. Sort the remaining patients in chronological order based on their admission day. In case there is a tie, they are sorted based on the length of stay in ascending order. Any further ties are arbitrarily resolved.

4. Repeat the following for each patient $p \in P \setminus O$ using the sorted order:

   (a) Determine for each room $r \in R$ whether the patient can be assigned to this room. This is done by checking

   - whether the room is compatible with patient p, i.e., $r \in R(p)$.
   - whether there is still capacity in the room on admission day d_{adm} to add this patient p, i.e.,

   $$f(r, d_{\text{adm}}) + m(r, d_{\text{adm}}) < C(r). \tag{5.2}$$

   Note that because we are scheduling in chronological order, we only need to check the admission day.

   This gives us a set of feasible rooms that the patient can be assigned to.

   (b) Sort these feasible rooms based on several criteria. The details for this can be found in Section 5.2.3.1.

   (c) Assign the patient to the first room of the list and update $f(r, d)$ and $m(r, d)$ accordingly.

   (d) If a patient has no feasible rooms, then the algorithm backtracks to a patient with at least two feasible rooms and tries the other one. Any changes to $f(r, d)$ and $m(r, d)$ are reverted.

5. If every patient has been assigned a room, the algorithm terminates.

Note that this algorithm is guaranteed to yield a feasible PR-assignment as it effectively enumerates every room choice for each patient until it finds a feasible solution. However, there is no guarantee that this is done efficiently.

5.2.2 Nurse-to-room heuristic

We can use the patient-to-room assignment from the previous section to derive a nurse-to-room assignment. Using a fixed patient-to-room assignment, the optimal nurse-to-room assignment can be found by solving an ILP adapted from Formulation (3.27). The main change from Formulation (3.27) is that variables and constraints directly related to the patient-to-room assignment are removed, which removes Constraints (3.5)-(3.8). Additionally, Constraint (3.10) could be simplified.

If we let $r(p)$ denote the room patient $p \in P$ was assigned to, then we would instead have

$$z_{nps} = x_{nr(p)s}, \quad \forall n \in N,\, p \in P,\, s \in S(n,p). \tag{5.3}$$

This states that nurse $n \in N$ is assigned to patient $p \in P$ during shift $s \in S(n,p)$ if and only if the nurse is assigned to room $r(p)$. Furthermore, Constraint (3.11) can be removed as it is implied by Constraints (3.9) and (5.3). The adapted ILP is given in Formulation (A.9). Although fixing the patient-to-room assignment significantly reduces the computation times, for large problem instances, it is not able to quickly produce a solution. Because of this, we provide a simplified algorithm.

The idea behind the algorithm is that, instead of considering all shifts simultaneously, the shifts are evaluated sequentially. For each shift $s \in S$, a simple ILP can determine the nurse assignment which leads to the lowest increase in objective value. Note that, for the skill violations (**S3**), workload violations (**S4**) and workload imbalance (**S5**) objective components, each shift can be optimized separately to find the optimal solution, as these are independent across shifts when the patient assignment is fixed. However, optimizing the continuity of care objective (**S1**) depends on choices in other shifts, which means that something must be done to account for this. In particular, suppose we are currently determining the nurse assignment in a shift $s \in S$. Let $\mathcal{N}_s(p)$ be the set of different nurses that were assigned to patient $p \in P(s)$ in shifts scheduled before s. The following constraints can then be added:

$$a_{np} = \begin{cases} 1, & \text{if } n \in \mathcal{N}_s(p), \\ z_{nps}, & \text{otherwise,} \end{cases} \quad \forall n \in N(s),\, p \in P(s). \tag{5.4}$$

This allows the ILP to optimize the continuity of care objective in a singular shift based on past decisions. This ILP can be found in Formulation (A.10) and is solved sequentially for each $s \in S$. Note that the specific order that these shifts are considered changes in the final solution. This explored more in Section 5.2.3.2.

Because this heuristic tries to solve the PNRA problem sequentially, we refer to this as the sequential greedy heuristic. If instead we use Formulation (A.9), where each shift is optimized simultaneously, we refer to it as the non-sequential greedy heuristic. Note that in that case it is still a heuristic as the patient-to-room assignment was heuristically obtained.

A drawback of the sequential heuristic is that the resulting nurse-to-room assignment is not unique. Within a single shift, there might be multiple optimal nurse-to-room assignments. If a different assignment is chosen during one of the first shifts, this choice could affect choices in later assignments, which might lead to a different final objective value. Although on a single machine, Gurobi always yields the same solution for a fixed seed, this solution might be different on another machine, which then also yields a different objective value. For a fairer evaluation, we always report averages over several runs with different seeds.

5.2.3 Improving the greedy heuristics

In Sections 5.2.1 and 5.2.2, two heuristics were described that can be used to find a solution to the PNRA problem quickly. However, there are two design decisions that we can change to try to improve the resulting solutions.

5.2.3.1 Room ordering

In the patient-to-room assignment heuristic, patients are assigned to rooms sequentially. For each patient $p \in P \setminus O$, a set of feasible room assignments is created based on the remaining capacity in each room and the set of incompatible rooms $R(p)$. These rooms are then sorted and the patient is assigned to the first room. The way these rooms is sorted is the most important part of the algorithm.

In the PNRA problem, there is only a single objective component directly related to the patient-to-room assignment, namely the gender-mixing constraint (**S2**). The goal is to sort the rooms of a patient such that we minimize this constraint, while also allowing for easier optimization for the other objective components (**S1**, **S3**, **S4**, **S5**). We consider three different sorting criteria:

- Gender (G): sort based on the number of days of a patient's stay that the room is labeled with the opposite gender. This means that, e.g. for a female patient $p \in P_F$, we sort the rooms ascendingly based on the value

$$\sum_{d \in D(p)} \mathbb{1}\left\{m(r,d) \geq 1\right\}, \tag{5.5}$$

  where $D(p)$ is the set of days that patient p is present, $\mathbb{1}$ is the indicator function and $m(r,d)$ is the number of male patients assigned to room $r \in R$ on day $d \in D$. In case the patient is male, $m(r,d)$ is replaced with $f(r,d)$, which instead counts the number of female patients. This criteria tries to greedily assign each patient to the room that leads to the lowest gender-mixing constraint violation.

- Occupancy level (O): sort based on the total number of patients in the room for each day of a patient's stay. This means that, e.g. for a patient $p \in P$, we sort the rooms ascendingly based on the value

$$\sum_{d \in D(p)} f(r,d) + m(r,d), \tag{5.6}$$

  where $D(p)$ is the set of days that patient p is present and $f(r,d) + m(r,d)$ is the sum of male and female patients assigned to room $r \in R$ on day $d \in D$. This criteria tries to greedily assign patients to the room with the lowest number of patients.

- Similarity score (S_α): sort the rooms based on how similar the patients in the room are to the current patient p, based on the continuity of care lower bound obtained in Section 4.2.1. Let $\mathcal{N}(p)$ denote the set cover of nurses obtained for patient $p \in P$, i.e., $\mathcal{N}(p)$ is the smallest set of nurses required to cover all shifts of patient p. Between two patients p_1 and p_2, we define

$$q(p_1,p_2) = \frac{|\mathcal{N}(p_1) \cap \mathcal{N}(p_2)|}{\min\left\{|\mathcal{N}(p_1)|, |\mathcal{N}(p_2)|\right\}}. \tag{5.7}$$

This gives a measure of how similar the two sets $\mathcal{N}(p_1)$ and $\mathcal{N}(p_2)$ are, or more precisely how much the smaller set also has nurses in the bigger set. Note that there can be multiple optimal set covers for each patient and this value therefore depends on the choice of this set $\mathcal{N}(p)$. To reduce this effect, we find for each patient p five different such set covers. The value $q(p_1, p_2)$ is then found by taking the maximum value over all set cover combinations.
To use this patient similarity, first define $P(p, r)$ to be the set of patients that were assigned to room r during the stay of patient p. The similarity score of a room is then defined as

$$\text{sim}_\alpha(p, r) = \begin{cases} \frac{1}{|P(p,r)|} \sum_{p' \in P(p,r)} q(p, p'), & \text{if } |P(p, r)| \neq 0, \\ \alpha, & \text{otherwise,} \end{cases} \tag{5.8}$$

where $\alpha \in [0, 1]$ is the default value a room is given when it is empty during a patient's stay. The rooms are sorted on this similarity score in descending order.

These three room ordering criteria can be combined to create different sorting rules. These are denoted with a string using the letters G, O and S_α, which refer to the sorting criteria. For example, the sorting rule GOS_0 refers to the rule in which we first sort the rooms based on the gender criterion, then using the occupancy criterion in case of ties and finally using the similarity criterion with default value $\alpha = 0$.

To determine which sorting rule is best, we consider the following rules: OG, GO, GS_0O, $GS_{0.5}O$, $GS_{0.8}O$, GS_1O, GOS_0. These were selected based on preliminary tests. Note that the value of α does not matter in the case of GOS_α, since an empty room is already preferred by the occupancy criterion. The sorting rules are evaluated using Formulation (A.9), which finds the optimal nurse-to-room assignment when a patient-to-room assignment is fixed. We set a maximum runtime of five minutes. As this is difficult to solve, the medium instances spend 150 seconds using the NoRel heuristic and the large instances spend the full five minutes using the NoRel heuristic. The resulting objective values for the training instances can be found in Table B.2. In Figure 5.2, the objective scores are plotted for each instance size class.

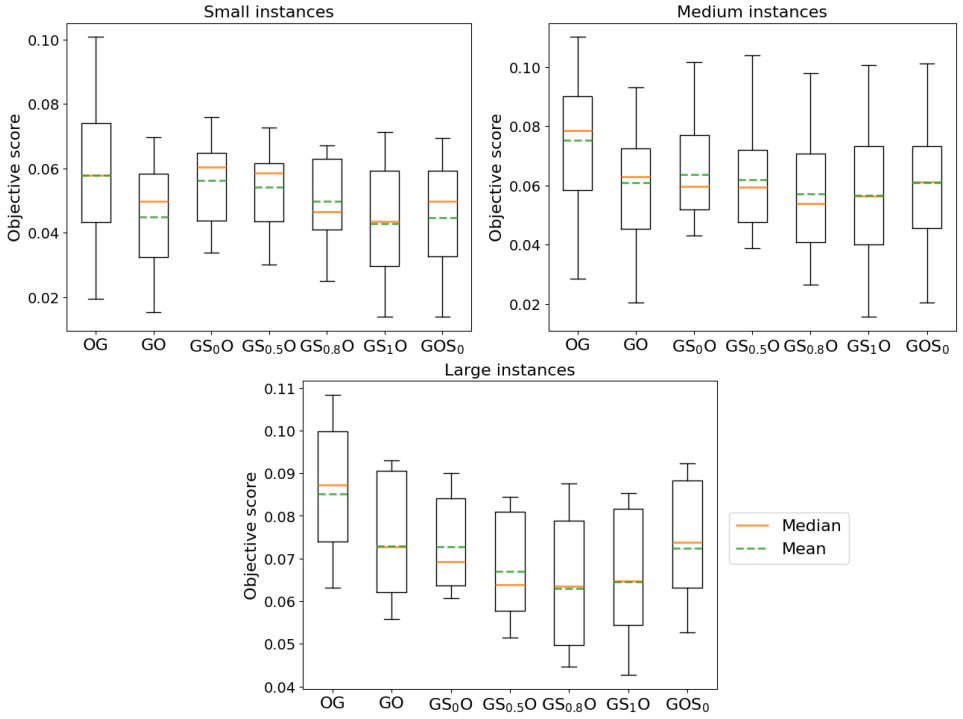


Figure 5.2: Objective scores for different room orderings. Results are separated into the three instance size classes.

For the small instances, the GS_1O sorting rule has both the lowest mean and median objective score. For the large instances, the same is true for sorting rule $GS_{0.8}O$. For medium instances, $GS_{0.8}O$ has a better median score, while GS_1O has a better average score. Because the averages for both sorting rules are very similar, we prefer the one with the lower median score. This means that we use $GS_{0.8}O$ for medium-sized instances. Regardless, we see that adding information about continuity of care in the patient-to-room assignment improves the solution compared to rules GO and OG.

The reason why larger instances prefer a slightly smaller value of α might be explained by the magnitude of the continuity of care objective value compared to other components. When $\alpha = 1$, the algorithm prefers to place new patients in empty rooms over rooms with at least one other patient. However, when $\alpha = 0.8$, a patient can be assigned to a non-empty room while another room is empty if the similarity score is high enough. In that case, the algorithm focuses more on improving continuity of care compared to other objective components. The reason that this is better for larger instances is because an increase in the number of patients leads to a larger increase in continuity of care objective g_{cc} compared to other objective components. This means that it is more important for larger instances to prioritize continuity of care, which is the case when α is smaller.

The value of α should also not be set too low. If, for example, $\alpha = 0$, then any occupied room with non-zero similarity score is preferred. This means that some rooms are left unoccupied. As the workload in that case is less spread across the rooms, this leads to worse values for the workload violations g_{wv} and imbalance g_{wi}.

5.2.3.2 Shift ordering

In the nurse-to-room assignment heuristic, nurses are assigned to rooms in each shift separately using an ILP, while taking into account the nurse assignment of previously assigned shifts. The most natural order to do this is to consider the shifts in chronological order. However, any arbitrary ordering could be used. To determine if there is another ordering that yields better solutions, we consider the following orderings:

- The chronological ordering (O_{Chrono}).
- Ordering based on the number of nurses in each shift (O_{Nurses}).
- Ordering based on the number of patients in each shift (O_{Patients}).
- Ordering based on the total workload of all patients in each shift (O_{Load}).
- Ordering based on the average workload of all patients in each shift (O_{Avg}).

For each order, we consider an ascending and a descending order. This is specified with a plus (ascending) or a minus (descending). So for example, reverse chronological ordering is denoted by O^{-}_{Chrono}. In the case of a tie between two shifts, they are always evaluated in chronological order.

To test the performance of each of these orderings, they were tested on the training instances. The patient-to-room assignment was created using the greedy patient-to-room assignment heuristic (5.2.1) using the appropriate room ordering determined in Section 5.2.3.1. Due to randomness in the sequential nurse-to-room assignment heuristic, the average results over ten runs for each instance are reported. In Table B.3, the average objective values can be found. In Figure 5.3, the average objective scores are plotted for each instance size class.

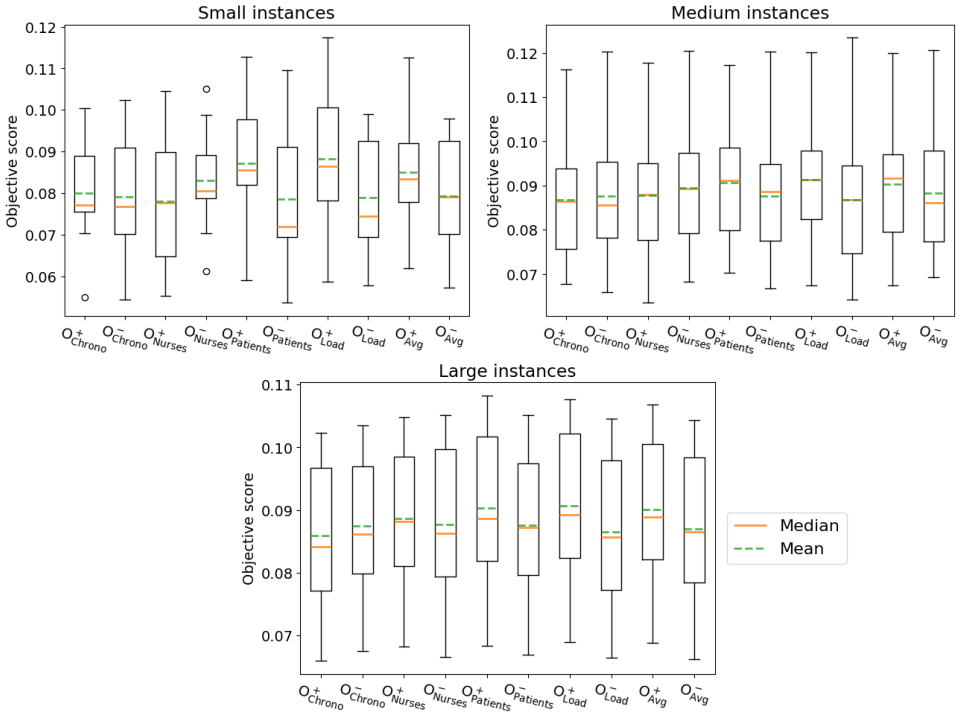


Figure 5.3: Objective scores for different shift orderings. A "-" in front of the room ordering indicates a descending order. Results are separated into the three instance size classes.

When looking at all results, we see that, on average, using the descending order for the number of patients, total workload and average workload is better than the ascending order. This means that it is generally better to schedule busier shift first.

A reason why this might be the case is that the continuity of care is not taken into account during the first shift that is scheduled. Regardless of the nurse assignment, the continuity of care objective is the same, namely $|P(s)|$. Therefore, the ILP fully focuses on minimizing the other objective components. Since those component values are higher in busier shifts, it is good to schedule those first.

This is shown for instance test01 in Figure 5.4, which shows how the objective value grows as more shifts are scheduled. The plot shows the results for a single run for each ordering. We see that all orderings that schedule busy shifts first (dashed lines) start with a large increase in the first few scheduled shifts, but decrease later on. After all shifts are scheduled, they achieve lower objective values.

Looking again at Figure 5.3, we see that, for the small instances, the lowest median score is obtained by the $\mathrm{O}^{-}_{\mathrm{Patients}}$ ordering, while the lowest mean score is achieved by the $\mathrm{O}^{+}_{\mathrm{Nurses}}$ ordering. Since the mean score is very similar in both cases, we prefer using the number of patients.

For the medium and large instances, the results differ less across the different orderings. Because of this, any improvement an ordering makes over another is partly due to randomness. For both the medium and large instances, we choose to simply use the chronological ordering as it performs quite well in most cases.

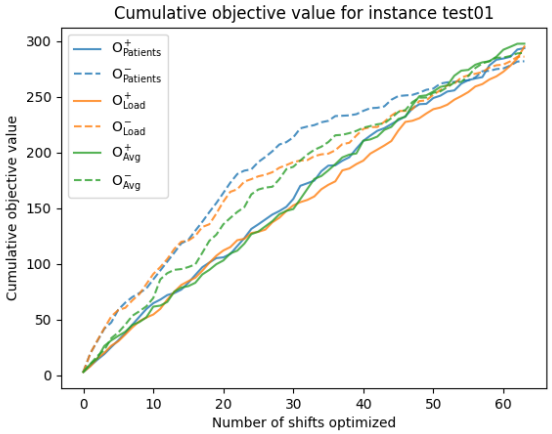


Figure 5.4: Cumulative objective value for instance test01 when using different shift orderings.

5.2.4 Computational results

In Table 5.1, the objective value obtained using the two greedy heuristics is shown for the ten test instances. Since the sequential nurse-to-room assignment is non-deterministic, the mean objective and computation time over ten runs is given. The non-sequential nurse assignment aims to find the optimal assignment for a fixed patient-to-room assignment and, therefore, gives an idea of how well the sequential heuristic performs. The runtime limit for the ILP used for this was set to five minutes. Note that the times reported in Table 5.1 could exceed this five minute limit since it includes the time needed to create the ILP. Also note that, similar to Section B.2, medium instances spent 2.5 minutes using the NoRel heuristic, while large instances spent the full five minutes using the NoRel heuristic.

Table 5.1: Greedy heuristic results. A "*" next to the instance name indicates that the non-sequential nurse assignment was terminated after five minutes.

Instance	Patient-to-room	Nurse-to-room			
		Sequential		Non-sequential	
	Runtime (s)	Objective	Runtime (s)	Objective	Runtime (s)
test01	0.224	279.332	0.317	257.217	6.332
test02	0.252	412.007	0.396	372.583	79.392
test03	0.181	165.107	0.262	142.633	0.302
test04*	0.418	434.395	0.548	382.033	300.277
test05	0.237	213.592	0.348	180.633	5.182
test06*	0.681	540.088	0.834	473.167	300.488
test07*	0.651	943.687	0.887	861.733	300.292
test08*	0.830	707.925	1.183	622.683	300.328
test09*	1.217	961.818	1.452	802.033	301.109
test10*	7.674	3562.593	21.827	3062.883	313.109

We see that the sequential greedy heuristic is able to find a feasible solution very quickly. For most instances, a solution is found within a couple of seconds. For large instance test10, however, this takes around 30 seconds, which is mostly spent on

the nurse assignment. The solution found using the non-sequential nurse-to-room assignment takes longer, but achieves a better objective value. This is logical as the non-sequential heuristic solves an ILP that aims to find the optimal nurse-to-room assignment for a given patient-to-room assignment, whereas the sequential heuristic makes simplifying assumptions to speed up computations.

These two heuristics can also be compared against the ILP from Chapter 3. Recall that in Section 4.3, the ILP was run for a maximum of 8 hours for the ten test instances. Table 5.2 shows how long it took for those runs to reach a better objective than those presented in Table 5.1.

Table 5.2: Time (s) needed for ILP Formulation (3.27) to find a better solution than the greedy heuristics. A "-" indicates that the ILP never found a better solution within 8 hours.

Instance	Sequential	Non-sequential
test01	3.456	13.667
test02	17.091	682.635
test03	3.754	12.350
test04	348.588	-
test05	3.691	11.250
test06	354.501	12374.594
test07	391.803	8693.451
test08	924.446	-
test09	4087.752	-
test10	6981.538	-

We see that, even for the sequential heuristic, the ILP can take much longer to reach the same objective value. This is especially true for instance test09 and test10, where it takes between one and two hours to improve.

For the non-sequential heuristic, the time needed to improve the solution becomes even larger and in some cases, the heuristic is able to find a better solution than the ILP. This happens mostly for the larger instances. For the smaller instances such as test01, test03 and test05, it took less than 15 seconds for the ILP to improve the value from Table 5.1. Since the non-sequential nurse assignment was optimal in those cases, a different patient-to-room assignment is needed to find a better solution.

5.3 Interval heuristic

This heuristic is inspired by instance test03. In this instance, there is a single day in which there are no patients. Because of this, the days before and the days after this day could be independently optimized, which might be leveraged by an ILP solver to speed up the solving procedure. For the other instances, we cannot find such a split. Regardless of how the days are split, there is at least one patient that is present during both sets of days. Despite this, we can still use this idea to find a heuristic solution. This heuristic works as follows:

1. Divide the planning horizon D into several intervals. We denote this as $\mathcal{I} = (L_1, \ldots, L_K)$, where each L_k denotes the number of days in the kth interval.

2. For each interval $k = 1, \ldots, K$, solve an ILP to find the patient-to-room and

nurse-to-room assignment that is optimal for that interval, while taking the assignments made in the previous intervals into account.

Because we split the days into several intervals, we refer to this heuristic as the interval heuristic.

The ILP used in this heuristic is adapted from Formulation (3.27). To highlight the changes, we consider intervals $\mathcal{I} = (L_1, \ldots, L_K)$. For each interval k, an ILP is solved. Because each interval considers only a subset of shifts, it also only considers a subset of patients and nurses. For interval k, let S^k denote the set of shifts for each day in that interval, the patients and nurses within an interval are then denoted with P^k and N^k, i.e.,

$$P^k = \bigcup_{s \in S^k} P(s), \tag{5.9}$$

$$N^k = \bigcup_{s \in S^k} N(s). \tag{5.10}$$

Furthermore, let P^0 be the set of occupants O. Then, instead of Constraints (3.8), we can use a more general set of constraints stating that the previously assigned patients must be assigned to the same room $r_{k-1}(p)$ as before, i.e.,

$$y_{pr_{k-1}(p)} = 1, \quad \forall p \in P^{k-1} \cap P^k. \tag{5.11}$$

Here, the set $P^{k-1} \cap P^k$ consists of all patients that were present during intervals k and $k-1$.

Additionally, to account for the continuity of care of the previous assignments, we add another set of constraints. Similar to the sequential nurse assignment heuristic, for each patient, we keep track of the set of assigned nurses. We use $\mathcal{N}_k(p)$ to denote the set of nurses that was assigned to patient $p \in P$ before interval k. For $k = 1$, this set is empty. After each interval, the set is updated according to the nurse-to-patient assignment. If a nurse $n \in N^k$ was assigned to patient $p \in P^k$ in a previous interval, i.e., if $n \in \mathcal{N}_k(p)$, then we add constraint

$$a_{np} = 1, \tag{5.12}$$

otherwise, we keep the Constraints (3.12) and (3.13), but adapt it to only use shifts in the current interval. This ILP is given in Formulation (A.11). Note that anytime a superscript k is used, it means that only the shifts or patients from interval k are used. For example, we define

$$S^k(n,p) := S^k \cap S(n,p). \tag{5.13}$$

The sets S^k_{early}, $S^k_{\text{early}}(p)$, $S^k(n)$, $S^k(p)$, $S^k(n,p)$, P^k_F and P^k_M are similarly defined.

This heuristic has three drawbacks. Firstly, when an interval is very short, there could be multiple optimal solutions for that interval. Due to how Gurobi solves ILPs, it might find a different solution based on the hardware on which Gurobi is running, even if the seed is fixed. For a fairer evaluation, we always report averages over several runs with different seeds.

The second problem is the solving speed. Generally, we want each interval to be as large as possible such that it is as close to the original problem as possible. However, as the interval length grows, the time it takes to solve each ILP grows

much more rapidly. This is especially true for larger instances. If we want a fast solution for larger instances, this means that each interval must be short, which leads to poorer solution quality.

Lastly, this heuristic is not guaranteed to find a feasible solution. This is shown in Example 5.3.1.

Example 5.3.1. *Consider a problem instance spanning two days. The instance consists of:*

- *Two nurses n_1 and n_2 present during both days.*
- *Three patients p_1, p_2 and p_3. Patients p_1 and p_2 are present during both days, while patient p_3 is only present on the second day.*
- *Two rooms r_1 and r_2. Room r_1 has capacity 1 and room r_2 has capacity 2. Room r_2 is incompatible for patient p_3.*

For simplicity, we assume that there is only a single shift on each day. For this example, the precise workload and skill requirement is not important, so we assume that it is the same for each patient. Similarly, the maximum workload and skill level is the same for each nurse.

If we use the interval heuristic with intervals of a single day, this could lead to an infeasible solution. When optimizing the workload imbalance on the first day, patients p_1 and p_2 are be assigned to different rooms and different nurses. This means that on day 2, room r_1 is fully occupied and patient p_3 has no compatible room available, which leads to infeasibility.

Despite these drawbacks, we can still use this heuristic to find solutions. In Section 5.3.1, we explain how the interval lengths are determined and how these are tuned. In Section 5.3.2, the heuristic is run and the results are shown.

5.3.1 Experimental setup

The interval heuristic has several design choices, namely choosing how the days are split into intervals and choosing how much runtime each interval has. Generally, splits with longer intervals produce better results, but also take longer to finish. Furthermore, larger instances also need more time to solve each ILP. For this reason, different instance size might perform better for different interval lengths. This is determined in Section 5.3.2. In particular, we determine the maximum interval length $L_{\max}$ for each size. Based on this value, the interval lengths are determined in the following way:

1. Define the number of intervals K based on $L_{\max}$. This is the number of times $L_{\max}$ can divide into the number of days $|D|$, rounded up, i.e.,

$$K = \left\lceil \frac{|D|}{L_{\max}} \right\rceil. \tag{5.14}$$

   Let δ_k denote the number of remaining days at the start of interval k, so $\delta_1 = |D|$.

2. Set $L_1 = \min\{\delta_1, L_{\max}\}$. This ensures that at least one interval has length $L_{\max}$, if there are enough days. Update the remaining number of days such that $\delta_2 = \delta_1 - L_1$

3. The remaining days are divided across the remaining intervals as equally as possible. In particular, for each $k \in \{2, \ldots, K\}$, we set

$$L_k = \left\lceil \frac{\delta_k}{K - k + 1} \right\rceil \tag{5.15}$$

and update $\delta_{k+1} = \delta_k - L_k$.

4. This yields interval lengths $\mathcal{I} = (L_1, \ldots, L_k)$.

So if we would have, for example, $|D| = 14$ and $L_{\max} = 3$, then the interval lengths would be $(3, 3, 3, 3, 2)$.

There are two reasons for creating the interval lengths in this manner. Firstly, the fact that we define $L_{\max}$ instead of K directly is that this makes comparisons for instances with the same size easier. Not all instances of the same size also have the same number of days. If we would set $K = 2$, then for an instance of two weeks this would give intervals of length 7, but for four weeks it would give intervals of length 14. This seems like an unfair comparison as increasing interval length has more effect on the runtime than the number of intervals.

Secondly, it might seem odd why we do not distribute the days roughly evenly across the K intervals, but instead assign at least one to be length $L_{\max}$. This is mostly to ensure that setting a different value for $L_{\max}$ also gives different interval lengths. For example, with $|D| = 14$ and $L_{\max} = 6$, the intervals could more equally be distributed into intervals $(5, 5, 4)$, but this would then be the same as $L_{\max} = 5$. Instead we choose to use $(6, 4, 4)$. For smaller values of $L_{\max}$ this is often not a problem and the two approaches would find the same interval lengths.

Although we have defined the interval lengths for a given $L_{\max}$, we have not selected the order of these lengths. Consider again the interval lengths of $(3, 3, 3, 3, 2)$. For these interval lengths, there are 5 different ways to order them. We must therefore choose an appropriate permutation of the interval lengths.

Recall instance test03. In this instance, there is a single day in which no patients are present. Because of this, we could choose a splits such that no patient is in more than one interval. Since these intervals are independent, solving them separately yields the optimal solution. This motivates choosing a permutation such that the number of patients present during multiple intervals is small. In particular, we choose the split $\mathcal{I}$ that minimizes the patient overlap score

$$\omega(\mathcal{I}) = \sum_{k=2}^{K} |P^{k-1} \cap P^k|, \tag{5.16}$$

where each term in the sum counts the number of patients present during both interval k and $k - 1$. The best split can then be found by iterating over all permutations and selecting the one with the lowest patient overlap score. Note that, for a given value $L_{\max}$, the number of permutations is quite low and they can be enumerated quickly.

The full procedure to determine the interval lengths for a given $L_{\max}$ is summarized in Algorithm 1.

Algorithm 1 Determining interval lengths

Require: D, $L_{\max}$
  $K \leftarrow \lceil |D|/L_{\max} \rceil$
  $\mathcal{I}$ is an empty list
  $\delta_1 = |D|$
  $k \leftarrow 1$
  while $k \leq K$ **do**
    if $k = 1$ **then**
      $L_k \leftarrow \min\{\delta_k, L_{\max}\}$ ▷ At least one interval of length $L_{\max}$
    else
      $L_k \leftarrow \lceil \delta_k/(K-k+1) \rceil$ ▷ Divide days evenly over remaining intervals
    end if
    $\delta_{k+1} \leftarrow \delta_k - L_k$
    add L_k to $\mathcal{I}$
    $k \leftarrow k + 1$
  end while
  $\mathcal{I} \leftarrow$ permutation of $\mathcal{I}$ with lowest overlap score $\omega(\mathcal{I})$
  return $\mathcal{I}$

Another important part of the interval heuristic is defining the runtime limits for the ILP in each interval. Because we try different interval lengths and number of intervals, we do not define a runtime limit for each interval directly, but instead use a total runtime budget. This budget is evenly divided across the different intervals. As longer planning horizons require more runtime, the budget depends on the number of weeks in the planning horizon. The instances in our dataset are either two, three or four weeks. Each week is given 2.5 minutes of runtime. So, an instance of two weeks has 5 minutes and one of four weeks has 10 minutes of runtime budget.

The budget is divided evenly across the intervals. If the ILP in an interval exceeds its allocated runtime limit, it terminates early and continues to the next interval with the best solution found at that time. If the ILP finishes before the time limit, the remaining budget is evenly divided across the remaining intervals. This ensures that the total runtime of the ILPs does not exceed the runtime budget, but intervals that are easy to solve do not waste a large portion of the runtime budget. Note that the runtime budget only limits the solving time of the ILPs, the creation of the ILP models is not considered in this budget.

Lastly, due to the randomness of the interval heuristic, any computational test is run ten times and the average objective value is reported. This eliminates some of the randomness and makes evaluations fairer.

5.3.2 Computational results

In Figure 5.5, the results for the experiment described in Section 5.3.1 are shown. For each instance size, three different plots are shown: the objective score, the percentage of the runtime budget used and the timeout rate. The timeout rate is the percentage of intervals that could not be solved within the allowed time limit and were timed out. Note that the heuristic was always able to find a solution within the time limit and never became infeasible due to earlier decisions.

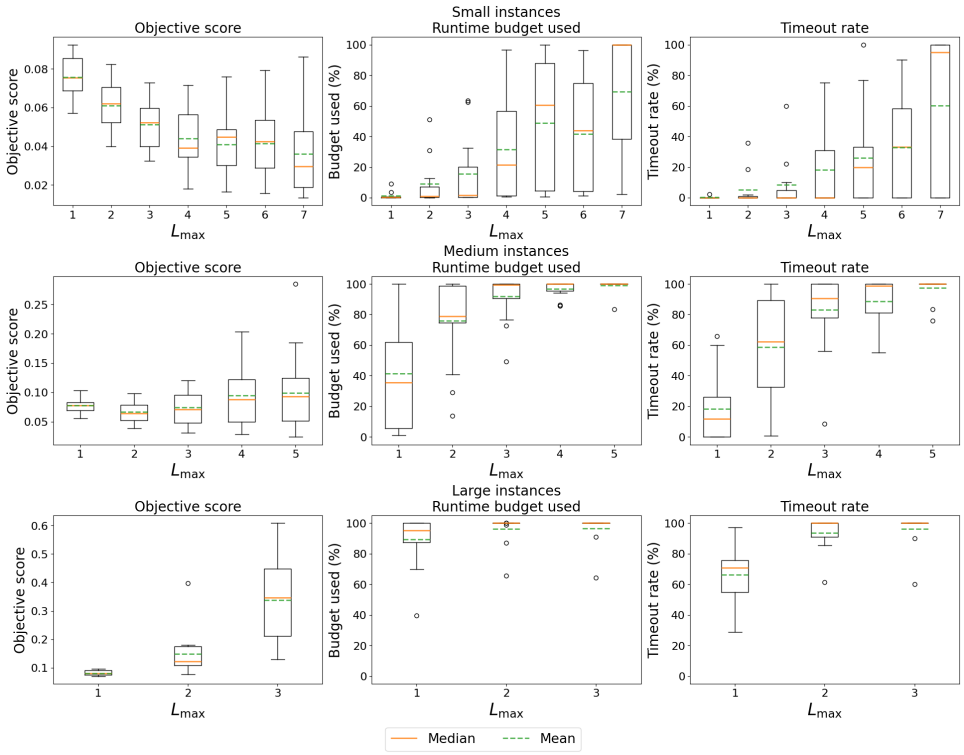


Figure 5.5: Results for the interval heuristic.

For all instance sizes, we clearly see that as we increase $L_{\max}$, the percentage of the runtime budget that is used increases. The time needed to solve each interval grows more rapidly than the number of intervals decreases when increasing $L_{\max}$.

For the small instances, on average, the objective score improves as $L_{\max}$ increases. This makes sense, as there are less intervals which also means that less patients are carried over. This means that less of an error is being made in the continuity of care optimization. For the small instances, we use $L_{\max} = 7$.

For the large instances, there is again a clear pattern, although completely opposite of the small instances. Here we see that the objective score rapidly increases as $L_{\max}$ increases. Even for $L_{\max} = 1$, the average timeout rate is 66.3%. For larger values of $L_{\max}$, the timeout becomes even higher and is often 100%. This highlights that, for large instances, the ILP in each interval could not be solved sufficiently. In each interval, a sub-optimal solution is selected and used for the next interval, which leads to a final solution with a low quality. This problem becomes even worse when $L_{\max}$ is increased. For the large instances, it is best to just keep $L_{\max} = 1$.

The medium instances lie somewhere in between. The objective does not clearly improve or worsen as $L_{\max}$ increases. Based on Figure 5.5, the best value for $L_{\max}$ in terms of average objective score is attained for $L_{\max} = 2$.

In Table 5.3, the results from the interval heuristic with these values for $L_{\max}$ are given for the ten test instances. Note that the mean runtime also includes the ILP model creation, which means that it could exceed the runtime budget. This happens for example with instance test10 which has a runtime budget of 7.5 minutes but runs for almost 9.5 minutes. The final column reports the time needed for ILP

Formulation (3.27) to find a solution with a better objective value than the average interval heuristic value, using the run from Section 4.3.

Table 5.3: Interval heuristic results.

Instance	$L_{\max}$	Objective	Runtime (s)	Time for ILP improvement (s)
test01	7	243.753	52.375	24.332
test02	7	363.965	300.899	1572.860
test03	7	142.450	194.885	29.595
test04	7	415.487	302.137	968.031
test05	7	177.215	301.743	30.375
test06	2	475.327	234.815	11768.130
test07	2	872.990	63.386	2583.763
test08	2	634.853	186.591	6292.433
test09	2	861.315	364.509	8047.653
test10	1	3931.072	569.761	2175.501

We see that for most of these instances, the solution obtained using the interval heuristic performs pretty well. Although they are never able to improve upon the solution found using the ILP formulation in Section 4.3, a solution of the same quality often takes much longer to find for the ILP.

5.4 Simulated annealing

Simulated annealing is a probabilistic local search metaheuristic named by Kirkpatrick et al. (1983). Simulated annealing has already been successfully used for the IHTC (Zanazzo, Smet, et al., 2025) and the IPNRA problem (Zanazzo, Ceschia, & Schaerf, 2025) in the past. It would therefore also be a good fit for the integrated problem we consider.

In general, simulated annealing starts by initializing a solution Q_0 and temperature T_0. Then, during each iteration k, a candidate solution is randomly selected from the neighborhood around the current solution. If this solution has a better objective value, this solution is accepted. If the solution is worse, this solution is only accepted with a certain probability that depends on the change in objective and the current temperature. After each iteration, the temperature is reduced, which reduces the likelihood that a worse solution is accepted.

There are several aspects to the simulated annealing algorithm that differ across different problems. In Section 5.4.1, the design of the simulated annealing algorithm is explained. In Section 5.4.2, the simulated annealing algorithm is written out. The algorithm contains several parameters, which are tuned in Section 5.4.3. Lastly, the algorithm is ran for the test instances and the results are shown in Section 5.4.4. In these sections, we introduce new notation which might differ slightly in meaning from notation used earlier in this thesis. The definition for each symbol is given in Table 5.4.

Table 5.4: Various symbols used for simulated annealing.

Symbol	Description
k	Iteration index. Starts at $k = 0$.
K	Total number of iterations.
Q_k	Solution at the start of iteration k.
Q_{best}	Best incumbent solution.
T_k	Temperature at the start of iteration k.
α	Cooling rate.
g	Objective function.
$\mu(Q)$	Function that denotes a move. Given a solution Q, returns a new solution that applies that move.
θ	Probability vector used to determine from which neighborhood a move is selected.
$\mathcal{N}_k(p)$	Set of all nurses assigned to patient $p \in P$ during their stay in solution Q_k.
$r_k(p)$	Room that patient $p \in P$ is assigned to in solution Q_k.

5.4.1 Algorithm design

When designing a simulated annealing heuristic, several aspects much be chosen: the initial solution, the candidate selection and the cooling schedule.

5.4.1.1 Initial Solution

For the initial solution, we could either create a randomized solution or use a solution found by one of the other heuristics. During the tuning procedure, we use randomized solutions such that the choice of the parameters does not depend on the quality of the starting solution.

The way we create a randomized solution is quite simple. For each patient $p \in P \setminus O$, we randomly select a room r from the set $R(p)$, i.e., the set of all compatible rooms for patient p. The occupants $p \in O$ are simply placed in the correct room $r_{\text{prev}}(p)$. Note that this randomized assignment might result in a patient-to-room assignment that violates the capacity constraints. However, similar to Zanazzo, Ceschia, and Schaerf (2025), we allow solutions that violate hard capacity constraints (**H1**) to "enchance the connectivity within the search space". A penalty is added to the objective function for each patient that exceeds a room's capacity. To ensure the algorithm prefers solutions that are actually feasible, the penalty is 100 times the largest weight of an objective component. During both the tuning and evaluation, all runs yielded feasible solutions.

The nurse-to-room assignment is not subject to any difficult hard constraints, so creating a random assignment is generally easy. The only requirement is that each room $r \in R$ has a nurse assigned to it during every shift $s \in S$. This can be done by, for each room $r \in R$ and shift $s \in S$, randomly selecting one nurse $n \in N(s)$.

5.4.1.2 Candidate selection

Within a simulated annealing algorithm, the solution changes by randomly selecting a new candidate solution Q'_k in the neighborhood around the current solution Q_k. This neighborhood is often generated by applying some type of move to the current

solution. In our simulated annealing algorithm, we consider four different move types:

- **MovePatient (MP)**: move a patient $p \in P \setminus O$ to a different room $r \in R(p)$ that is compatible for that patient. A move of this type is selected by first randomly selecting a patient $p \in P \setminus O$ with $|R(p)| \geq 2$ and then selecting a room from the set $R(p) \setminus \{r_k(p)\}$, where $r_k(p)$ denotes the room assigned to patient p in current solution Q_k.
- **SwapPatients (SP)**: swap the rooms of two patients that are assigned to different rooms. Note that both rooms must be compatible for both patients. Furthermore, we only consider patient pairs that have at least one overlapping shift. A move of this type is selected by randomly selecting a pair (p_1, p_2) from the set
$$\begin{aligned}\{(p_1, p_2) : p_1,\, p_2 \in P \setminus O,\ p_1 \neq p_2,\\ r_k(p_1) \in R(p_2),\ r_k(p_2) \in R(p_1),\\ r_k(p_1) \neq r(p_2)\}.\end{aligned} \tag{5.17}$$
- **ChangeNurse (CN)**: change the nurse that is assigned to a room during a particular shift. A move of this type is selected by first randomly selecting a room $r \in R$ and shift $s \in S$ where $|N(s)| \geq 2$. If n_1 is the nurse currently assigned to room r during shift s, then a new nurse n_2 is randomly selected from the set $N(s) \setminus \{n_1\}$.
- **RemoveNurse (RN)**: remove a nurse $n \in N$ from a patient p's set of assigned nurses $\mathcal{N}_k(p)$. This is done by replacing them with another nurse from $\mathcal{N}_k(p)$ for each shift they were assigned. A move of this type is selected by first randomly selecting a pair (p, n), where each shift that nurse $n \in \mathcal{N}_k(p)$ is assigned to patient $p \in P$, the nurse could be replaced with a different nurse $n' \in \mathcal{N}_k(p)$. Note that n' does not need to be the same for each shift. After selecting such a pair (p, n), for each shift s that nurse n is assigned to patient p, we randomly select a nurse n' from the set $\mathcal{N}_k(p) \cap N(s) \setminus \{n\}$, i.e. the set of other nurses present during shift s that was already assigned to patient p during another shift.

Out of all of these moves, the RemoveNurse move is probably most complicated. The idea behind this move is that it can reduce the continuity of care value for a single patient in a single move. If a nurse $n \in \mathcal{N}_k(p)$ is replaced with another nurse from $\mathcal{N}_k(p)$ for each shift that they were assigned to patient p, then that nurse is removed from $\mathcal{N}_k(p)$ while no other nurses were added. This reduces the continuity of care value for patient p by 1.

Within each iteration, we randomly select a candidate solution Q'_k by first randomly selecting a move type and then randomly selecting a move of that type. The probability that a MovePatient, SwapPatients, ChangeNurse or RemoveNurse move is selected is denoted using θ_{MP}, θ_{SP}, θ_{CN} and θ_{RN}, respectively. These values are parameters to the problem and are tuned in Section 5.4.3. Since these values represent probabilities, we must ensure that each value is non-negative and that

$$\theta_{\text{MP}} + \theta_{\text{SP}} + \theta_{\text{CN}} + \theta_{\text{RN}} = 1. \tag{5.18}$$

The probability vector is denoted using

$$\theta = (\theta_{\text{MP}},\, \theta_{\text{SP}},\, \theta_{\text{CN}},\, \theta_{\text{RN}})^T. \tag{5.19}$$

Due to how the RemoveNurse move is defined, it is often the case that there are no feasible RemoveNurse moves in an iteration. In that case, a new move is drawn within the same iteration using probability vector θ', where

$$\theta'_{\text{MP}} = \frac{\theta_{\text{MP}}}{1 - \theta_{\text{RN}}}, \tag{5.20}$$

$$\theta'_{\text{SP}} = \frac{\theta_{\text{SP}}}{1 - \theta_{\text{RN}}}, \tag{5.21}$$

$$\theta'_{\text{CN}} = \frac{\theta_{\text{CN}}}{1 - \theta_{\text{RN}}}, \tag{5.22}$$

$$\theta'_{\text{RN}} = 0. \tag{5.23}$$

This ensures that the ratio in probability between two different moves remains the same. Apart from the RemoveNurse moves, the other move types could also have no feasible moves. However, this is very uncommon or unrealistic. For example, the only way for there to be no MovePatient moves is if every patient $p \in P$ has $|R(p)| = 1$, i.e., if the patient-to-room assignment is fixed. We generally assume that this is not the case. However, if this is the case, a new move can be sampled in a similar way as for the RemoveNurse moves. The same can be done for the SwapPatients and ChangeNurse moves.

5.4.1.3 Cooling schedule

We use a geometric cooling scheme. This means that the temperature is reduced after each iteration by multiplying the current temperature with a cooling rate α, i.e.,

$$T_{k+1} = \alpha T_k. \tag{5.24}$$

Instead of defining a fixed cooling rate, we define one relative to the number of iterations. This allows us to use a different number of iterations for the tuning and evaluation. Let T_0 be the initial temperature and T_K be the final temperature after the last iteration. The cooling rate is then defined as

$$\alpha = \sqrt[K]{\frac{T_K}{T_0}}. \tag{5.25}$$

Note that, since the iterations start with $k = 0$, the temperature T_K is the temperature after it was updated in the final iteration. This is therefore not the temperature used in the final iteration.

5.4.2 Full algorithm

Using the notation from Table 5.4, the simulated annealing works as follows:

1. Initialize a solution Q_0 and temperature T_0. Also, set the best incumbent solution $Q_{\text{best}} = Q_0$.
2. For $k = 0, \ldots, K - 1$ do the following:
   (a) Randomly select a move type (MovePatient, SwapPatients, ChangeNurse or RemoveNurse) according to the probability vector θ.
   (b) In the selected move type, choose an appropriate move μ (how this is done differs per move and is described in Section 5.4.1.2).

(c) Create candidate solution Q'_k by applying move μ, i.e., $Q'_k = \mu(Q_k)$.

(d) Determine how the objective changes and define

$$\Delta_k = g(Q'_k) - g(Q_k). \tag{5.26}$$

(e) Sample $U \sim \text{Unif}(0, 1)$.

(f) If $U \leq \exp(-\Delta_k/T_k)$, accept the solution and set $Q_{k+1} = Q'_k$. Otherwise, we set $Q_{k+1} = Q_k$. Note that in the case that $\Delta_k \leq 0$, the move is always accepted.

(g) Update the temperature $T_{k+1} = \alpha T_k$.

(h) If the objective value of the new solution is better than the best incumbent solution, i.e., if $g(Q_{k+1}) < g(Q_{\text{best}})$, then the best incumbent solution is updated $Q_{\text{best}} = Q_{k+1}$

3. After performing K iterations, the best incumbent solution Q_{best} is returned.

One detail that is important to the simulated annealing algorithm is how the change in objective Δ_k is computed. You can simply compute the full objective value again for the candidate solution and compare against the old objective value, but this is computationally expensive. As we are only changing a small part of the solution, this can be done more efficiently. The details of this can be found in the implementation on Github[1].

5.4.3 Parameter tuning

Before we can evaluate the simulated annealing algorithm, we must decide on appropriate values for the various parameters. We fix the number of iterations $K = 1{,}000{,}000$. This value was selected since this is enough iterations for the algorithm to converge to a solution. This is illustrated in Figure 5.6, which shows the objective value of the current and best incumbent solution at every iteration for a single run of instance test01. The parameters used are those found at the end of the tuning procedure. We see that it indeed converges to a solution.

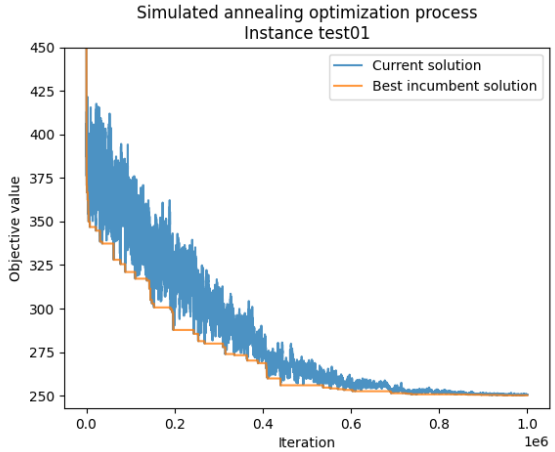


Figure 5.6: A run of simulated annealing for instance test01. It shows the objective value for the current and best incumbent solution in every iteration.

[1] Available at https://github.com/LarsdeHoop/PNRA-problem.

With the number of iterations fixed, the only parameters that must be tuned are the initial temperature T_0, the final temperature T_K and the four probabilities θ_{MP}, θ_{SP}, θ_{CN} and θ_{RN}. For each parameter, we select several values that are tested. We test all combinations of these values. Note that several parameter combinations are disregarded. This happens when $T_0 \leq T_k$, to ensure the temperature decreases, and when the probabilities do not sum to 1, to ensure θ is a probability vector.

Since simulated annealing is non-deterministic, the algorithm is run ten times with each parameter combination. Due to the large computation time required to run the algorithm several times for each parameter combination, the experiments are performed using DelftBlue (DHPC, 2024) to reduce the total computation time. This is a high performance computer that allows several job scripts to run simultaneously. However, since DelftBlue has a 24 hour limit for each submitted job, the number of different parameter combinations in each round of tuning could not be too high. For this reason, we often consider only three or four options for each parameter.

In the following sections, three rounds of parameter tuning are discussed. During the first round, we try different values for each parameter to get a better understanding of what parameter combinations perform well and which ones do not. This is done by looking at the average objective score over all instances of each instance size class. The objective score is defined in Section 5.1. During the second round, we search closer around the best parameter combination found during the first round. In particular, we focus more on tuning the move probabilities. In the last round, we fix these probabilities and try many different values for T_0 and T_K.

5.4.3.1 Round 1

During a preliminary test, reasonable values for the various parameters were determined. In particular, we observed that the value for θ_{CN} should be higher than the other probabilities. This is most likely because moving a patient to another room changes more about the solution and the objective value. If a patient is moved, more ChangeNurse moves are required to "repair" the solution.

The values that are tested in the first tuning round are shown in Table 5.5. Note that the same values are considered for each instance size.

Table 5.5: Parameter options (round 1).

Parameter	Values
T_0	$\{0.1, 1, 10, 100\}$
T_K	$\{0.001, 0.01, 0.1\}$
θ_{MP}	$\{0.1, 0.2, 0.3\}$
θ_{SP}	$\{0, 0.1, 0.2, 0.3\}$
θ_{CN}	$\{0.5, 0.6, 0.7\}$
θ_{RN}	$\{0, 0.1, 0.2, 0.3\}$

After running each feasible parameter combination 10 times, several observations can be made. In Figure 5.7, the mean objective score over all runs are shown for the different instance sizes. Each point represents the mean objective score across all instances and runs for a single parameter combination for one instance size. In this figure, we focus on the parameters T_0 and θ_{SP}. The x-axis only separates the parameter combinations based on the value for T_0. Different points with the same value T_0 are ordered arbitrarily. The color of each point corresponds to the value for θ_{SP}.

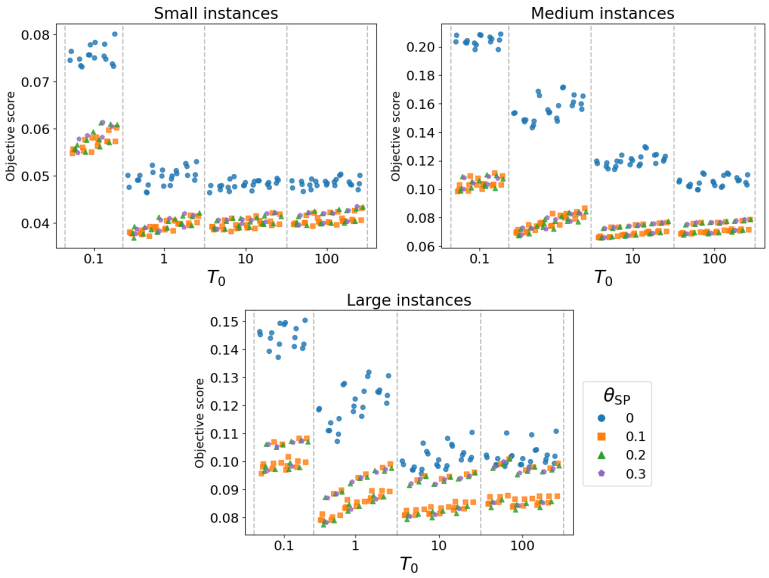


Figure 5.7: Simulated annealing tuning results (round 1) focused on parameters T_0 and θ_{SP}. The x-axis is split based on T_0 and the coloring is based on θ_{SP}. Parameter combinations where the value of T_0 is the same are ordered arbitrarily.

From this figure, it is easy to see that combinations with $T_0 = 0.1$ or $\theta_{\text{SP}} = 0$ generally have worse outcomes across all instance size classes. In Figure 5.8, these combinations are omitted and instead of T_0 and θ_{SP}, parameters T_K and θ_{RN} are shown.

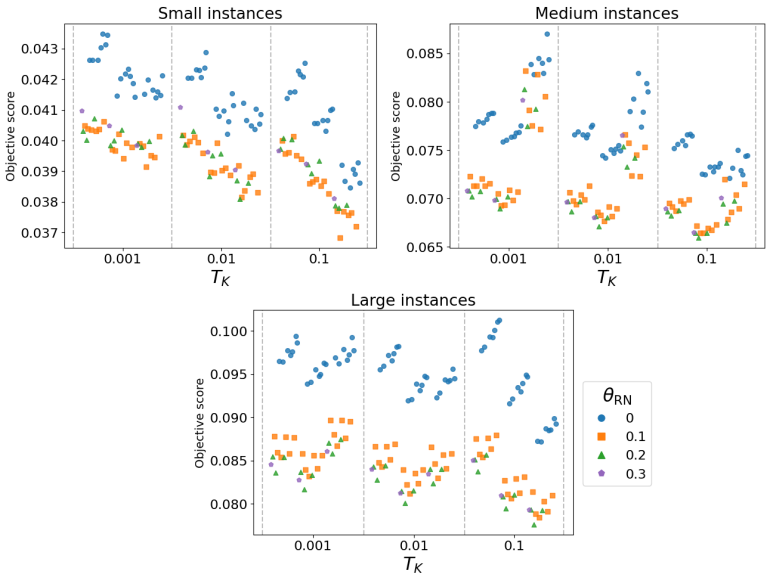


Figure 5.8: Simulated annealing tuning results (round 1) focused on parameters T_K and θ_{RN}. The x-axis is split based on T_K and the coloring is based on θ_{RN}. Within each value of T_K the ordering of points is arbitrary. Parameter combinations where $T_0 = 0.1$ or $\theta_{\text{SP}} = 0$ are omitted.

We see that, for all instance sizes, the results for parameter combinations where $\theta_{\text{RN}} = 0$ are clearly worse. This indicates that the addition of the RemoveNurse move type has a positive effect on the objective value. Additionally, we see that a higher value for T_K generally leads to better scores for all instance sizes. For the other two parameters, θ_{MP} and θ_{CN}, we could not see such clear patterns and they are therefore not plotted.

In Table 5.6, the best parameter combination for each instance size is given. This is the combination that achieved the lowest objective score averaged over all instances of that size. They are very similar for each instance size, however, the most important difference is the value for T_0 for medium instances, which is higher.

Table 5.6: Best parameter combination per instance size (round 1).

Instance size	T_0	T_K	θ_{MP}	θ_{SP}	θ_{CN}	θ_{RN}	Avg. score
Small	1	0.1	0.1	0.2	0.6	0.1	0.03683
Medium	10	0.1	0.1	0.2	0.5	0.2	0.06595
Large	1	0.1	0.1	0.2	0.5	0.2	0.07766

5.4.3.2 Round 2

Based on the results from the first tuning round, we determine new parameter options for each instance size separately. The following parameter values are tried for each parameter:

1. The value of the parameter in the best parameter combination (see Table 5.6).
2. An intermediate value both below and above this value. This is often taken as the midpoint between this value and an adjacent value from the previously considered values (see Table 5.5).
3. For each probability parameter, we consider an additional value. This is the value that appeared most often in the top 10 of parameter combinations, apart from the value of the parameter in the best parameter combination. This value is added as it might become better in combination with newly considered parameter values. For medium and large instances, this was only done for parameters θ_{MP} and θ_{CN} to stay within the DelftBlue computation time limit.

The resulting options for each instance size are shown in Table 5.7.

Table 5.7: Parameter options per instance size (round 2).

Parameter	Values per instance size		
	Small	Medium	Large
T_0	$\{0.5, 1, 5\}$	$\{5, 10, 50\}$	$\{0.5, 1, 5\}$
T_K	$\{0.05, 0.1, 0.5\}$	$\{0.05, 0.1, 0.5\}$	$\{0.05, 0.1, 0.5\}$
θ_{MP}	$\{0.05, 0.1, 0.15, 0.2\}$	$\{0.05, 0.1, 0.15, 0.2\}$	$\{0.05, 0.1, 0.15, 0.2\}$
θ_{SP}	$\{0.1, 0.15, 0.2, 0.25\}$	$\{0.15, 0.2, 0.25\}$	$\{0.15, 0.2, 0.25\}$
θ_{CN}	$\{0.5, 0.55, 0.60, 0.65\}$	$\{0.45, 0.5, 0.55, 0.6\}$	$\{0.45, 0.5, 0.55, 0.6\}$
θ_{RN}	$\{0.05, 0.1, 0.15, 0.2\}$	$\{0.15, 0.2, 0.25\}$	$\{0.15, 0.2, 0.25\}$

In Figure 5.7, the results from running all feasible parameter combinations from Table 5.7 are shown. Again, each point represents the mean objective score for a

single parameter combination. We first focus on the values for T_0 and T_K. Note that the medium instances tried different values for T_0 compared to the small and large instances. For this reason there are also more feasible parameter combinations. Because we want $T_0 > T_K$, we remove combinations where $T_0 = T_K = 0.5$ for the small and large instances.

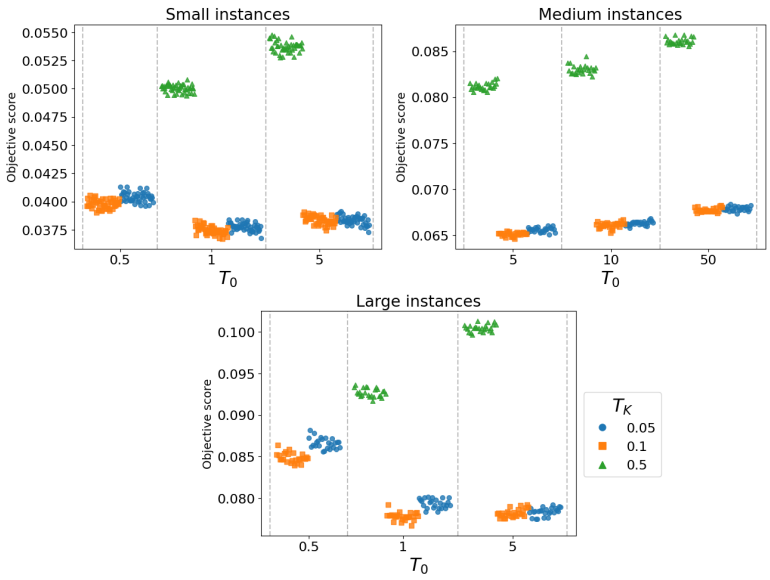


Figure 5.9: Simulated annealing tuning results (round 2) focused on parameters T_0 and T_K. The x-axis is split based on T_0 and the coloring is based on T_K. Within each value of T_0 the ordering of points is arbitrary.

Firstly, consider parameter T_K. We can clearly see that, across all instance sizes, parameter combinations with $T_K = 0.5$ produce significantly worse results. We can also see that combinations with $T_K = 0.1$ seem to perform slightly better than with $T_K = 0.05$.

For parameter T_0, we can clearly see that, for the small and large instances, combinations with $T_0 = 0.5$ generally perform worse, where the difference is larger for the large instances. The other two values have more similar scores, but combinations with $T_0 = 1$ have slightly better scores for small instances. For large instances, it depends on the value of T_k which of the two values of T_0 performs better. For medium instances, the worst results are attained when $T_0 = 50$ and the best results are attained when $T_0 = 5$.

For the other parameters, it is more difficult to see any patterns from such images. In Figure 5.10, several boxplots can be seen. Each boxplot is made by looking at all parameter combinations where a certain parameter has a specific value. For example, the leftmost boxplot in the topleft subplot is made using all parameter combinations where $\theta_{\text{MP}} = 0.05$.

In these plots, the combinations which used the worst values for T_0 and T_K, as determined using Figure 5.9, were removed for each instance size. This means that combinations where $T_0 = 0.5$ or $T_K = 0.5$ were not included for small and large instances and that combinations where $T_0 = 50$ or $T_K = 0.5$ were not included for medium instances.

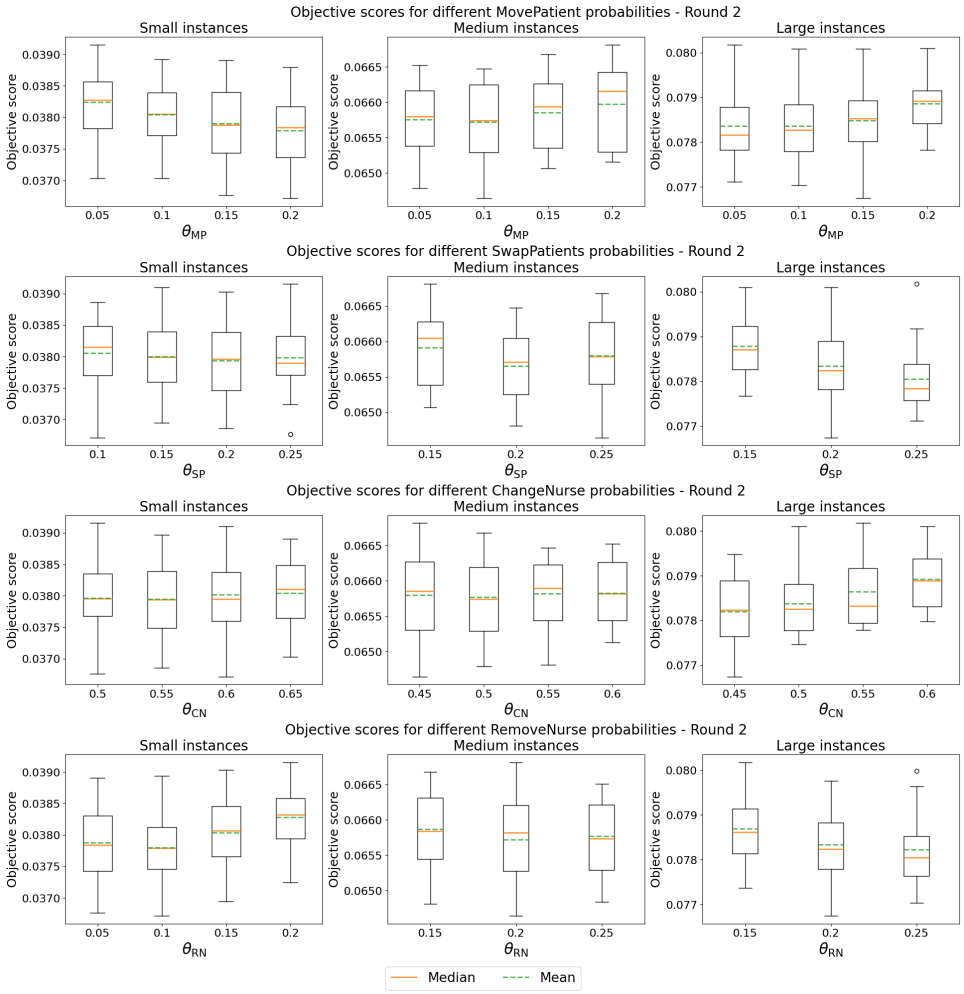


Figure 5.10: Simulated annealing tuning results (round 2) for the move probability parameters. For small and large instances, parameter combinations where $T_0 = 0.5$ or $T_K = 0.5$ were removed. For medium instances, parameter combinations where $T_0 = 50$ or $T_K = 0.5$ were removed.

Based on these figures, we see that there are no clear patterns that occur for all instance sizes. Additionally, on this scale, any difference between parameters could very likely be caused by randomness. Furthermore, these plots ignore the dependence that these parameters have. Regardless, we can still observe several things.

For θ_{MP}, there appears to be a slight downward trend for the small instances. On average, $\theta_{\text{MP}} = 0.2$ leads to better objective scores. The opposite seems true for the medium and large instances. There, $\theta_{\text{MP}} = 0.2$ has the worst objective scores and lower values are preferred. Instead, the larger instances seem to prefer higher values for θ_{SP}. This could suggest that one million iterations is not enough to sufficiently improve the solution for large instances. It might therefore prefer to make moves that more drastically change the solution, such as swapping patients instead of moving a single patient. The same behavior can be seen for the nurse related moves. Large instances prefer to use RemoveNurse moves more as this changes the nurse in more shifts simultaneously.

In Table 5.8, the parameter combination that yielded the best average objective score for each instance size is shown. The same pattern can be seen that small instances prefer moves that move a single patient or change a single nurse, while larger instances prefer more drastic moves.

Table 5.8: Best parameter combination per instance size (round 2).

Instance size	T_0	T_K	θ_{MP}	θ_{SP}	θ_{CN}	θ_{RN}	Avg. score
Small	1	0.1	0.2	0.1	0.6	0.1	0.03671
Medium	5	0.1	0.1	0.25	0.45	0.2	0.06464
Large	1	0.1	0.15	0.2	0.45	0.2	0.07674

5.4.3.3 Round 3

For the final round, we only focus on the temperature. During the previous rounds, we found that changing the temperature has more impact on the objective value than changing θ. For this reason, we use the θ values belonging to the best found parameter combination from the previous round. We try multiple different values for T_0 and T_K. These can be found in Table 5.9.

Table 5.9: Parameter options per instance size (round 3).

Instance size	T_0	T_K
Small	$\{0.6, 0.7, 0.8, 0.9, 1, 2, 3, 4\}$	$\{0.06, 0.07, 0.08, 0.09, 0.1, 0.2, 0.3, 0.4\}$
Medium	$\{2, 3, 4, 5, 6, 7, 8, 9\}$	$\{0.06, 0.07, 0.08, 0.09, 0.1, 0.2, 0.3, 0.4\}$
Large	$\{0.6, 0.7, 0.8, 0.9, 1, 2, 3, 4\}$	$\{0.06, 0.07, 0.08, 0.09, 0.1, 0.2, 0.3, 0.4\}$

The results for running these various choices for T_0 and T_K are shown in Figure 5.11.

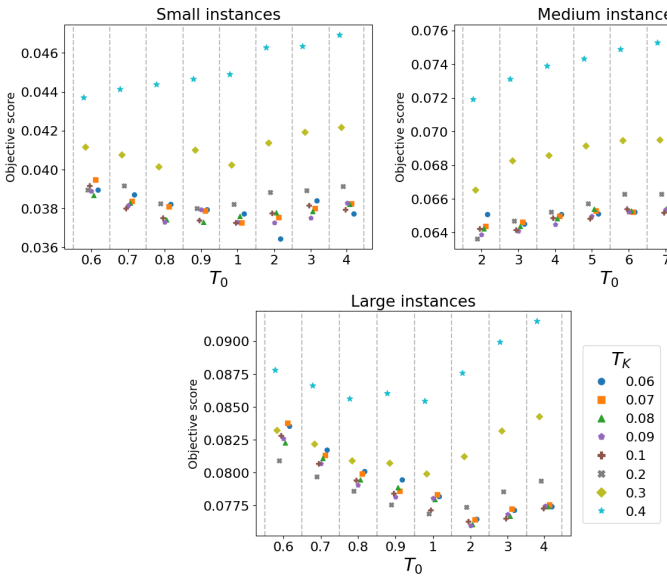


Figure 5.11: Simulated annealing tuning results (round 3) focused on parameters T_0 and T_K.

From this, it is clear that values $T_K = 0.3$ and $T_K = 0.4$ produce worse results for all instance sizes. Additionally, we see that, for all instance sizes, the lowest objective score is attained for $T_0 = 2$. In Table 5.10, the best combination of parameter values for each instance size class is given.

Table 5.10: Best parameter combination per instance size (round 3).

Instance size	T_0	T_K	θ_{MP}	θ_{SP}	θ_{CN}	θ_{RN}	Mean score
Small	2	0.06	0.2	0.1	0.6	0.1	0.03644
Medium	2	0.2	0.1	0.25	0.45	0.2	0.06363
Large	2	0.09	0.15	0.2	0.45	0.2	0.07600

5.4.4 Computational results

In Table 5.11, the results from simulated annealing with the parameters from Table 5.10 are given. Since simulated annealing is a random search algorithm, the results shown are averaged over ten runs. In each run, a new initial solution was randomly created using the procedure from Section 5.4.1.1. The final column reports the time needed for ILP Formulation (3.27) to find a solution with a better objective value than the average simulated annealing objective value, using the run from Section 4.3.

Table 5.11: Simulated annealing results. A "-" indicates that the ILP never found a better solution within 8 hours.

Instance	Objective	Runtime (s)	Time for ILP improvement (s)
test01	247.335	54.778	18.124
test02	358.835	55.399	3075.541
test03	144.335	55.558	4.298
test04	385.145	60.123	-
test05	181.445	57.660	11.250
test06	467.965	60.045	21934.894
test07	857.603	77.321	9408.895
test08	624.040	72.461	-
test09	852.985	73.524	14922.601
test10	3411.882	136.188	-

For most instances, performing one million iterations of simulated annealing takes roughly 1 minute. For the large instance test10, this is over two minutes. This is because larger instances have more moves for each type, take more time to calculate the change in objective for a specific move and have a larger solution that must be copied each time a better solution is found. This means that larger instances generally take more time per iteration.

Despite that, the objective values found are generally very good. Even though it only took roughly one or two minutes to complete, the simulated annealing algorithm is sometimes able to find better solutions than running ILP Formulation (3.27) for 8 hours. This happens for instances test04, test08 and test10. Additionally, for instances test06 and test09, it takes many hours for the ILP to improve upon this solution.

5.5 Comparison

The results from Tables 5.1, 5.3 and 5.11 are summarized in Table 5.12. This allows us to more easily compare the different heuristics.

Table 5.12: Objective values obtained using the different heuristics. Values are taken from Tables 5.1, 5.3 and 5.11.

Instance	Greedy (seq.)	Greedy (non-seq.)	Interval heuristic	Simulated annealing
test01	279.332	257.217	**243.753**	247.335
test02	412.007	372.583	363.965	**358.835**
test03	165.107	142.633	**142.450**	144.335
test04	434.395	**382.033**	415.487	385.145
test05	213.592	180.633	**177.215**	181.445
test06	540.088	473.167	475.327	**467.965**
test07	943.687	861.733	872.990	**857.603**
test08	707.925	**622.683**	634.853	624.040
test09	961.818	**802.033**	861.315	852.985
test10	3562.593	**3062.883**	3931.072	3411.882

Comparing the four heuristics, we clearly see that the greedy sequential heuristic has the worst performance out of all the heuristics. Only for instance test10 does the interval heuristic have a higher objective value. The greedy non-sequential heuristic performs much better, especially for larger instances. For the small instances, the heuristic is held back by the patient-to-room assignment. Instead the interval heuristic and simulated annealing perform better in those cases.

Note that the comparison made here is not entirely fair. The results for each heuristic were computed using very different runtimes. The greedy heuristics were given a runtime limit of five minutes, while the interval heuristic was given a runtime based on the length of the planning horizon. Additionally, all the results for simulated annealing were obtained using $K = 1{,}000{,}000$ iterations which often took between one and two minutes. Because of these differences in runtimes, it is an unfair comparison. In Chapter 6, we give each heuristic the same maximum runtime and evaluate them on the evaluation dataset defined in Section 5.1.

Chapter 6

Evaluation

In this chapter, the heuristics described in Chapter 5 are evaluated. Firstly, in Section 6.1, the heuristics are run for the evaluation instances and the results are discussed. Then, Section 6.2 analyses how the weights impact the solution.

6.1 Evaluation

In this section, we evaluate the heuristics introduced in Chapter 5. This is done on the evaluation dataset defined in Section 5.1. Recall that during the tuning of each of the heuristics, several different time limits were used. For the greedy heuristics, a time limit of five minutes was used. For the interval heuristic, a time limit was set based on the number of weeks in the planning horizon. For simulated annealing, one million iterations were performed regardless of runtime.

In order for us to more fairly evaluate each heuristic, we give each heuristic the same maximum runtime. In particular, we use the idea from the interval heuristic where the runtime is based on the length of the planning horizon. For each week, the heuristics have 2.5 minutes of runtime. For the greedy and interval heuristics, this limits only the solving time of the ILPs and not the model creation time. For smaller instances, this is often negligible. However, for larger instances, it can take more time.

For simulated annealing, using this runtime is slightly more work. As we only define the number of iterations K instead of a runtime limit, we approximate the number of iterations needed. To do this, the simulated annealing algorithm is first run for $K = 1{,}000{,}000$ iterations. If t is the runtime for this amount of iterations and t_{goal} is the desired runtime limit, then the new number of iterations is set to be

$$K = \lfloor 1{,}000{,}000 \cdot \frac{t_{\text{goal}}}{t} \rfloor \tag{6.1}$$

This assumes that each iteration roughly takes the same amount of time to calculate and computes the appropriate value for K based on that. Note that, since the cooling rate α depends on the initial temperature T_0, final temperature T_K and the number of iterations K, we can use the same parameters as determined in Section 5.4.3.

In Table 6.1, the results for the greedy heuristics are shown. Note that for instance m19, the patient-to-room assignment heuristic was not able to find a solution within a reasonable time. The algorithm was manually terminated after several hours since

there was no built-in runtime limit. The reason why this took so long is because instance m19 is a very busy instance. On almost every day, the rooms are at full capacity. This makes it harder to find a feasible room assignment. Additionally, there are 18 rooms available in the instance. This means that in each step of the algorithm, there are more options for a patient. Backtracking to an earlier patient assignment therefore takes much more time.

Table 6.1: Results for the greedy heuristic. A "-" indicates that the greedy heuristic found no solution within a reasonable time.

Instance	Patient-to-room	Nurse-to-room			
		Sequential		Non-sequential	
	Runtime (s)	Objective	Runtime (s)	Objective	Runtime (s)
m01	0.321	906.882	0.387	872.033	1.177
m02	0.280	323.045	0.548	279.950	95.615
m03	0.209	289.900	0.206	277.183	0.265
m04	0.429	471.347	0.501	434.700	300.227
m05	0.480	499.373	0.512	455.283	300.305
m06	1.059	924.947	1.886	781.167	600.747
m07	0.531	643.730	0.545	601.883	300.216
m08	1.621	1183.157	2.342	982.783	601.051
m09	1.859	2177.377	2.376	2012.333	601.035
m10	0.984	1154.470	1.400	1045.600	450.927
m11	2.409	1500.852	3.453	1154.883	602.164
m12	1.846	1686.765	2.909	1493.833	451.613
m13	2.574	1580.527	6.995	1301.000	456.523
m14	1.797	1338.422	2.966	1147.750	301.226
m16	1.867	1663.323	3.119	1506.150	302.102
m17	1.649	1512.165	2.429	1349.700	301.009
m19	-	-	-	-	-
m21	2.148	1986.157	4.083	1847.833	301.458
m15	3.587	3198.123	8.676	2989.917	605.619
m18	2.595	1917.645	7.759	1706.767	301.854
m20	2.651	1805.113	6.718	1503.717	302.389
m22	2.668	1568.878	14.084	1297.350	302.147
m23	3.726	2005.803	22.890	1772.217	306.685
m24	6.928	3154.407	30.384	2809.267	458.220
m25	3.864	2346.198	10.606	2130.933	454.214
m26	6.794	3827.528	21.617	3329.767	615.647
m27	2.767	2003.418	10.010	1753.700	303.972
m28	11.657	3838.658	77.545	3224.450	608.655
m29	6.788	3343.910	23.873	2608.267	611.303
m30	7.682	4203.528	50.064	3916.433	461.113

One of the differences between these results and those of Table 5.1, is that the runtimes are generally higher. For the non-sequential nurse assignment, this is to be expected as the runtime limit is also higher for most instances. The sequential nurse assignment, however, could previously be solved within a second or two, except for instance test10 which took roughly 20 seconds. Now, for the evaluation instances, it can sometimes take roughly a minute to find a solution. This could be due to the fact that these instances are, in general, larger than the instances used throughout training, which increases the runtime.

When comparing the results between the sequential and non-sequential nurse assignment, the same conclusion can be made as in Section 5.2.4. For all instances, the non-sequential assignment outperforms the sequential assignment. This is logical as the non-sequential assignment solves an ILP that tries to find the optimal nurse assignment for a fixed patient assignment, whereas the non-sequential assignment only tries to optimize the nurse assignment for each shift individually. The sequential solution is therefore part of the non-sequential ILP's solution space and it therefore finds a better solution, provided that the computation time is sufficiently large.

Next, the results for the evaluation instances of the interval heuristic are given in Table 6.2.

Table 6.2: Results for the interval heuristic.

Instance	Objective	Runtime (s)	Runtime budget (s)	Timeout rate (%)
m01	824.908	301.819	600	50.00
m02	274.118	600.495	600	25.00
m03	258.433	85.917	300	0.00
m04	429.198	300.945	300	100.00
m05	450.187	301.279	300	100.00
m06	841.562	253.293	600	17.86
m07	622.025	55.541	300	2.86
m08	1069.415	530.735	600	61.43
m09	2019.463	478.659	600	72.86
m10	1033.155	306.796	450	36.36
m11	1334.780	444.730	600	62.86
m12	1591.500	457.746	450	81.82
m13	1649.868	420.012	450	87.27
m14	1270.928	315.284	300	100.00
m16	1664.422	313.168	300	98.57
m17	1427.437	312.783	300	98.57
m19	1603.237	310.342	300	100.00
m21	2095.982	315.121	300	100.00
m15	3065.888	619.935	600	74.29
m18	1880.235	302.562	300	66.43
m20	1781.325	287.155	300	65.71
m22	1565.772	331.615	300	70.71
m23	2097.650	296.415	300	68.57
m24	3145.672	490.055	450	70.48
m25	2306.762	436.928	450	74.76
m26	3841.007	611.777	600	66.79
m27	1978.795	239.667	300	54.29
m28	4406.160	740.689	600	75.71
m29	3941.432	534.965	600	45.36
m30	6072.215	572.168	450	78.57

When looking at the mean runtime and budget for instance m01, it seems almost as if the wrong runtime limit was set. The average runtime seems much closer to five minutes, while the budget was 10 minutes. This could be explained by looking at the number of new arrivals in each interval. Note that, since instance m01 is a small instance, each interval consists of seven days. In the first interval, there are a total of 17 new patients arriving, which need to be assigned to a room. In the second interval, this is even higher at 28 new patients. In these first two intervals, the ILPs reach the runtime limit. In both of the remaining intervals, five new patients arrive. This makes the ILPs much easier as only a small number of patients need to be assigned to new rooms. The ILP for each of these two last intervals can be solved very quickly, which means that the runtime is mostly determined by the first two intervals. Since they both reach their runtime limit of 150 seconds, the total runtime is around 300 seconds.

Apart from instance m01, there are three other instances that stand out because of their runtimes. The first of these, m07, stands out because of the low runtime. Most other medium instances have a much larger runtime. This might suggest that instance m07 could handle longer intervals and maybe does not belong with the medium instances. In constrast, instances m28 and m30 have much larger runtimes. They both exceed their runtime budget with more than two minutes. This is caused mostly because the budget does not include the time needed to create each ILP. Although each interval only consists of a single day for these instances, it still takes a long time to create the model. These two instances are the largest of all the instances, so this is not surprising.

When comparing the solution quality with those from the greedy heuristic, we generally see that the interval heuristic performs better for smaller instances, while the greedy heuristic performs better for larger instances. This is generally because larger instances consider shorter intervals to prevent the ILPs from becoming too difficult. For large instances, each interval considers only a single day. Despite that, it still has trouble solving each individual ILP. This leads to sub-optimal solutions being selected and used for the next intervals.

In Table 6.3, the results for the simulated annealing heuristic are given.

Table 6.3: Results for simulated annealing

Instance	K	Objective	Runtime (s)	Runtime budget (s)
m01	10,271,089	835.427	602.685	600
m02	10,202,009	277.552	592.623	600
m03	5,529,816	260.672	317.440	300
m04	5,404,640	425.912	319.552	300
m05	5,199,085	437.525	315.259	300
m06	7,289,543	792.748	574.882	600
m07	4,872,798	617.645	322.200	300
m08	7,627,374	992.088	633.697	600
m09	7,268,657	1919.398	581.121	600
m10	6,392,651	1016.657	477.388	450
m11	6,854,462	1231.040	586.029	600
m12	5,551,440	1484.527	443.389	450
m13	4,928,426	1353.840	449.166	450
m14	3,919,420	1161.270	319.669	300
m16	3,792,408	1471.692	299.230	300
m17	4,151,479	1317.112	328.041	300
m19	3,888,633	1518.827	299.259	300
m21	3,848,966	1898.120	302.857	300
m15	6,295,362	3057.700	571.819	600
m18	3,743,338	1733.962	318.683	300
m20	3,418,675	1557.273	300.614	300
m22	3,421,643	1331.782	309.045	300
m23	3,172,343	1763.392	302.975	300
m24	3,605,884	2901.070	398.328	450
m25	4,651,607	2172.640	421.210	450
m26	4,425,129	3440.133	535.032	600
m27	3,481,687	1772.430	297.135	300
m28	3,582,765	3412.828	491.609	600
m29	4,039,533	2862.192	481.104	600
m30	3,248,726	4080.588	414.284	450

Observe that the method for approximating the runtime worked reasonably well. For some instances, it overestimated the runtime and for others the runtime was underestimated. The largest difference between the runtime and the runtime budget was for instance m29, where the real runtime was roughly two minutes below the runtime budget. This difference could have been reduced by initially using a value for K that was closer to the value presented in Table 6.3.

The results from Tables 6.1, 6.2 and 6.3 are summarized in Table B.4. These results are also visualized in Figure 6.1. It shows the objective score for the different heuristics. Recall that the objective score was defined in Section 5.1 and allows us to compare the objective value for different instances. Note that, since the greedy heuristic was unable to find a solution for instance m19, this instance is not included in the boxplot for both greedy heuristics for the medium instances.

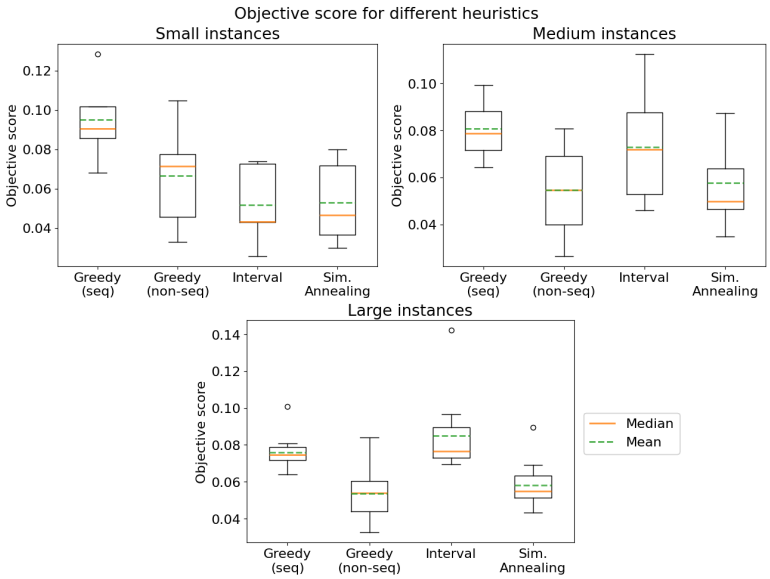


Figure 6.1: Results for all heuristics on the evaluation dataset.

When looking at the results for the small instances, we see that the interval heuristic and simulated annealing algorithms yield the best objective scores, with the interval heuristic having a slightly better mean and median score. The non-sequential greedy heuristic is often worse than these other heuristics. Since the nurse-to-room assignment was often solved within the time limit (see Table 6.1), a better patient-to-room assignment heuristic is needed to achieve better solutions.

For the medium and large instances, the interval heuristic is no longer the best. Due to the increased problem size, the heuristic is no longer able to sufficiently solve the ILPs, which leads to worse performance. This is especially clear for the large instances, where it is on average worse than the greedy sequential heuristic. For the medium instances, the best mean score is obtained using the greedy non-sequential heuristic, while the best median score is obtained using simulated annealing. For the large instances, the greedy non-sequential heuristic has both the best mean and median score.

6.2 Sensitivity analysis

In this section, we investigate what the impact of different weights is on the solution and how sensitive each objective component is to changing the weights.

Recall that simulated annealing accepts certain worsening moves with a probability based on how much the objective changes. This change in objective is denoted with Δ_k for iteration k. If we change the weights of the objective components, then this also changes how small or large Δ_k is. If, for example, all weights were set to $\frac{1}{100}$, then Δ_k would also become 100 times smaller. This also means that the algorithm is much more likely to accept this move, as worsening moves are accepted with probability $\exp(-\Delta_k/T_k)$. If we keep the same parameters for T_0 and T_K, the algorithm might not converge to a solution due to this high acceptance probability.

For this reason we must ensure that the weights remain of a similar size. In

particular, we assume that the weights are scaled such that

$$\beta_{\text{cc}} + \beta_{\text{gm}} + \beta_{\text{sv}} + \beta_{\text{wv}} + \beta_{\text{wi}} = 5. \tag{6.2}$$

Although a change in β still changes the value of Δ_k, this ensures that it has roughly the same order of magnitude. For the greedy and interval heuristics, the scale of the weights does not matter for performance, but we still assume they are normalized.

To investigate the impact of changing the weights, we look at instance m01. In Section 6.1, we determined that the best heuristic for this instance is the interval heuristic. We therefore use this heuristic to do the analysis. For each weight, we try values 0, 1, 2, 3, 4 and 5. The other weights are all set to the same value while ensuring Equation (6.2) is satisfied. For example, if we set $\beta_{\text{cc}} = 3$, then the other weights are all set to 0.5. This is done for each weight to see how this changes the obtained solution.

In order to display the results in a single figure, we must rescale the objective component values. This is because each objective component has very different ranges of possible values. For example, the continuity of care has a value around 300, while the gender-mixing component has a value around 10. Rescaling allows us to compare results more easily. We use a similar procedure as in Section 5.1. The main change is that we now consider the lower and upper bounds for the individual objective components instead of the weighted objective function. We call the obtained value the component score. The lower and upper bound needed to compute this can be obtained using the methods described in Sections 4.2.1 and 4.2.2. In particular, we use the partial relaxations to obtain these bounds. These are given for instance m01 in Table 6.4.

Table 6.4: Lower and upper bounds for each objective component for instance m01.

Type of bound	Continuity of care	Gender-mixing	Skill violations	Workload violations	Workload Imbalance
Lower bound	257	0	251	31	4.967
Upper bound	543	81	904	695	161.317

Using these bounds, we can see the impact that changing a singular weight has on the solution. In Figure 6.2, these results are shown. Note that we still use the same runtime budget of 10 minutes for the interval heuristic and average over ten runs.

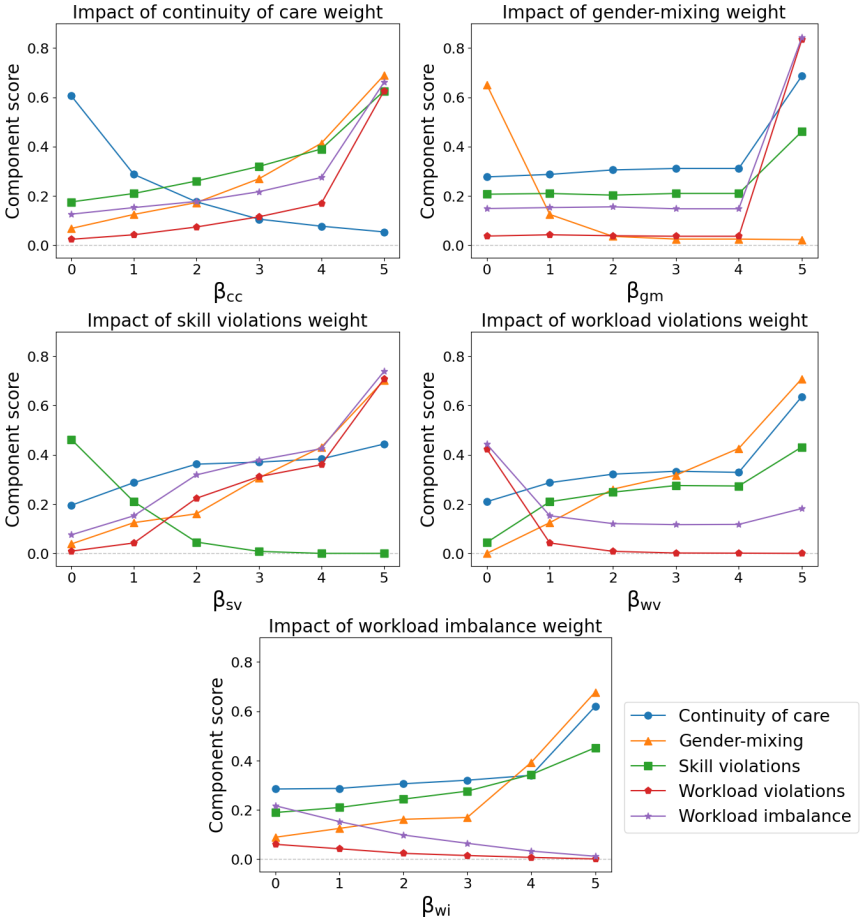


Figure 6.2: Results of sensitivity analysis. Each subplot shows how each objective component changes when a singular objective weight is changed.

For each objective component, we see that increasing its weight decreases the corresponding objective component value. This makes sense as more emphasis is placed on minimizing that objective component. Consequently, the other objective components then increase in value. There are several exceptions to this.

One of these exceptions is for the gender-mixing weight β_{gm}. When this weight is increased, the other components change very little. This could be explained by the fact that this instance is not very busy. On average, only 63.3% of beds are occupied. This means that it is easier to swap patients around to optimize the other objective components, while primarily minimizing the gender-mixing objective. Only when $\beta_{\text{gm}} = 5$, are the other components changed much. This is because the other components have weight 0 and are not taken into account at all.

Other exceptions are the workload violations and workload imbalance components. We see that, if one of these weights increases, both objective components decrease at the same time. This makes sense as both components aim to evenly distribute workload across nurses. Only when $\beta_{\text{wv}} = 5$ and the workload imbalance is no longer taken into account does the workload imbalance objective increase again. This is because, in that case, the model only wants to make sure each nurse has a workload below their maximum. It does not care whether a nurse is assigned

no workload at all during a shift. This is the main reason why this additional objective component was included in the problem formulation.

Another observation that can be made regarding the workload violation and workload imbalance components is when $\beta_{\text{wv}} = 0$. In that case, we see that, the workload violation component increases as it is no longer taken into account. However, the workload imbalance component also increases a lot. Since the workload imbalance objective has generally very small values compared to the other objective components, it prioritizes minimizing those other components to achieve a lower weighted objective value. Without the workload violations objective, the model does not have enough incentive to evenly distribute workload and instead minimizes the other objectives. This might indicate that we need both objectives or that we need to increase the workload imbalance weight relative to the other weights.

Overall, we see that changing the continuity of care and the skill violations weights has the most impact on the other objective components. The gender-mixing, workload violations and workload imbalance weights change the other objective components very little. Out of all objective components, the gender-mixing component seems to change most when other weights are changed.

Chapter 7

Online version of PNRA

In this chapter, we extend the problem by including emergency patients into the problem, which are not known to the planner beforehand. In Section 7.1, the updated problem is described, including the changes to the objective function. Then, Sections 7.2-7.4 describe the changes made to each of the heuristics to allow for use in the online setting. Lastly, these heuristic are tested on online instances in Section 7.5.

7.1 Updated problem description

In the original problem description from Section 2.2, we assumed that the problem is static and deterministic. We assumed that all patient information was known and that it could not change. In practice, however, this is not the case. For example, patients could arrive unexpectedly because of some emergency. To make the problem more realistic, we add these emergency patients to the problem.

We call a patient an emergency patient if they were not known to the planner beforehand and arrive unexpectedly. The day that a patient becomes known to the planner is called the registration day d_{reg}. Although we call them emergency patients, their registration day may be several days before they are admitted. This might be a patient that must be treated quickly, but is not in a life-threatening situation. Every time a new patient is registered, the schedule must be revised and changed to fit them in. It is therefore an online problem. When the schedule is updated, all previous decisions are fixed. So, for patients that were admitted before the current day, the room assignment is fixed. Similarly, the nurse-to-room assignments for previous days is also fixed. Other decisions that are made in the future can still be changed.

To more formally describe this online problem, we use some notation used in the ILP from Chapter 3. Although we do not formulate an ILP to solve the problem, this is useful to more accurately describe how things change compared to the offline version. Furthermore, these ideas are used in the ILPs for the online greedy and interval heuristics.

In the online version, several important symbols from Section 3.1 change slightly in meaning. Similar to the offline problem, we still consider a planning horizon of days D with corresponding shifts S. We refer to the first day of D as the current day. We assume that on this day, the schedule must be updated to fit in new emergency

patients. In particular, the set P consists of all patients that are registered on the current day. The set P can be split into three different sets O, P_{old} and P_{new}.

Similar to the offline setting, the set O denotes the current occupants. These patients were admitted before the current planning horizon and are therefore already assigned to a room. This room is denoted with $r_{\text{prev}}(p)$ for patient $p \in O$. One change compared to the offline setting is that we now take the previous nurse assignment into account. We let $\mathcal{N}_{\text{prev}}(p)$ denote the set of nurses that were previously assigned to patient $p \in O$. We use this set to compute the continuity of care for $p \in O$ within the current planning horizon. In terms of ILP constraints, this could be implemented using constraints

$$a_{np} = 1, \quad \forall p \in O,\, n \in \mathcal{N}_{\text{prev}}(p), \tag{7.1}$$

where a_{np} is defined in Section 3.2. This ensures that nurses that were previously assigned to a patient $p \in O$ already count towards the continuity of care value for that patient.

The sets P_{old} and P_{new} are new to the online problem. Unlike the occupants, these patients are admitted during the current time horizon and therefore are not already assigned to a room. The main difference between the two sets is that the patients in P_{old} where already known before the current time horizon. This means that they were included when a schedule was made in the past. For these patients, we therefore know the room assignment determined in that schedule. In particular, for $p \in P_{\text{old}}$ and $r \in R$, we know the value for patient-to-room assignment variable y_{pr}. These values can be used as a warm start for ILPs or for the initial solution in simulated annealing. The set P_{new} consists of patients that were newly registered on the current day. They were therefore not included in previous schedules.

Similar to how the patient-to-room assignment from the previous schedule can be used, we can also use the previous nurse-to-room assignment. In particular, let $S_{\text{prev}} \subseteq S$ be the shifts in S that were also considered during the previous schedule. We might have $S_{\text{prev}} \neq S$ when we consider a rolling horizon approach in which, for example, we always schedule 7 days ahead. This means that the previous schedule did not consider some of the days in D. For $s \in S_{\text{prev}}$, we know the value for nurse-to-room assignment variable x_{nrs} for $n \in N(s)$ and $r \in R$. Similar to the patient-to-room assignment variable, we can use this as a warm start for ILPs or for the initial solution in simulated annealing.

The OPNRA (Online Patient and Nurse to Room Assignment) problem can now be described. The goal is to find a feasible patient-to-room assignment and nurse-to-room assignment such that previous decisions are taken into account while also accounting for additional unknown patients arriving in the future. We still try to minimize the same objective components described in Section 2.2.3, i.e., continuity of care (**S1**), gender-mixing (**S2**), skill violations (**S3**), maximum workload violation (**S4**) and workload imbalance (**S5**). These values are minimized each time the schedule is updated. However, to evaluate a scheduling method, a longer time period is considered in which the schedule is updated several times.

Note that, similar to the interval heuristic from Section 5.3, a feasible instance can become infeasible when a choice is fixed. This is highlighted in Example 7.1.1.

Example 7.1.1. *We consider two nearly identical instances. Let both instances consist of two days and two rooms r_1 and r_2, which both have a single bed. In both instances, there is a known patient p_1 staying for two days. There is also an emergency patient p_2 arriving on the second day and only registering then. In each instance, this patient has a different incompatible room:*

- *In one instance, room r_1 is incompatible for patient p_2. To ensure feasibility, patient p_1 must be assigned to room r_1.*
- *In the other instance, room r_2 is incompatible for patient p_2. To ensure feasibility, patient p_1 must be assigned to room r_2.*

In an offline setting, both instances are feasible. Because, in that case, we know that patient p_2 cannot stay in one of the two rooms, we must assign patient p_1 to that room instead. In an online setting, this is not the case. Because patient p_2 is unknown to the planner on the first day, any (deterministic) algorithm makes the same assignment for patient p_1 in both instances. This means that one of the two instances becomes infeasible.

This example highlights that an instance feasible for the offline PNRA problem might become infeasible in an online setting depending on the choices made during earlier shifts. There are several ways to deal with this:

- *Soften incompatible room constraint*: if the incompatible room constraint is softened, the only remaining hard constraint regarding patient-to-room assignment is the capacity constraint. As long as the number of patients does not exceed the total capacity in each shift, the problem is feasible. This solution is, however, not ideal as this means that patients can stay in rooms that do not have the correct equipment needed.
- *Cancel/delay planned patients*: planned patients could be canceled or delayed until the problem is feasible. This would ideally not be done by an algorithm but instead by a planner with knowledge about the severity of the patient's condition to evaluate which patient could best be canceled.
- *Allow transfers*: originally, transfers were not included to simplify the problem as they were not needed for a feasible solution. If we allow transfers, the online problem remains feasible.

Out of these possible options, allowing transfers seems most realistic to include in our algorithms. As transfers are inconvenient for both patients and nurses, we want to minimize the number of room transfers as much as possible. This is therefore also included in the objective function with weight β_{tr}. We assume that transfers can only occur for the current occupants O. This means that transfers are not planned days in advance, but rather only occur when needed to produce a feasible solution. To implement this in an ILP, we could add binary variable v_p^{tr} for $p \in O$ and constraints

$$v_p^{\text{tr}} = 1 - y_{pr_{\text{prev}}(p)}, \quad \forall p \in O. \tag{7.2}$$

Adding v_p^{tr} for $p \in O$ with weight β_{tr} to the objective function ensures that transferring a patient to a different room is penalized. Because we only transfer current occupants, most heuristics from Chapter 5 need only small changes to work in the online setting. In particular, three changes can be made for all heuristics:

- Transfers must be allowed for current occupants.
- Continuity of care calculation must account for previous (fixed) nurse assignments, i.e., $\mathcal{N}_{\text{prev}}(p)$.
- Assignments from previous schedules can optionally be used for an initial solution.

How these changes are implemented in each of the heuristics, is discussed in Sections 7.2-7.4.

7.2 Online greedy heuristic

The greedy heuristic from Section 5.2 can be adapted into an online heuristic by making several changes.

Recall from Section 5.2.1 that the offline patient-to-room assignment heuristic starts with assigning current occupants to the correct room. However, because transfers are allowed in the online problem, this changes slightly. Instead, the current occupants are treated similarly as the other patients. The set of feasible rooms is determined and then sorted. The main change is that the first room in their feasible room list is always the room that they were assigned to previously. This ensures that transfers only occur when they are necessary for feasibility. Note that this does not take into account the weight β_{tr}, which is similar to how the gender-mixing weight β_{gm} was not included in both the online and offline heuristic.

Although this does allow for transfers, it might be very slow. If a transfer is needed to ensure feasibility, the algorithm first tries all options for the other patients before finally changing the room of an occupant. Because there are many possible room assignments, this might take very long.

The nurse-to-room assignment heuristic is even easier to change. Recall that, in the sequential heuristic, the set $\mathcal{N}_s(p)$ denoted the set of nurses that were assigned to patient $p \in P$ in shifts scheduled before shift s. Originally, this set was initialized as an empty set and was updated each time a new nurse was assigned to patient p. In the online version, we assume that each occupant $p \in O$ has a set $\mathcal{N}_{\text{prev}}(p)$ denoting the set of nurses that were assigned to them before the current shift. Initializing $\mathcal{N}_s(p) = \mathcal{N}_{\text{prev}}(p)$ for occupants takes into account the previous nurse assignments. The ILP is therefore the same as Formulation (A.10).

For the non-sequential heuristic, we can take this into account by replacing Constraint (3.12) for patients $p \in O$ with Constraint (7.1). This ensures that, for a patient $p \in O$, a nurse $n \in \mathcal{N}_{\text{prev}}(p)$ already counts towards the continuity of care objective. The ILP used in the online non-sequential greedy heuristic is given in Formulation (A.12). Note that Constraints (3.12) and (3.13) could be removed for $p \in O$ and $n \in \mathcal{N}_{\text{prev}}(p)$. This was however not done when writing out the ILP in Formulation (A.12) for simplicity. In the implementation, these constraints are removed.

Lastly, for both the sequential and non-sequential heuristics, we can use the previously made schedule. In particular, we initialize the values x_{nrs} for $s \in S_{\text{prev}}$, $n \in N(s)$, $r \in R$ using the previous values. This can reduce the computation time needed to solve the ILPs.

7.3 Online interval heuristic

The interval heuristic introduced in Section 5.3 can be adapted into an online heuristic by making several changes.

Due to the possible transfer of occupants O, Constraint (5.11) must be slightly changed. For the first interval $k = 1$, we add binary variable v_p^{tr} for $p \in O$ and change the constraint to be

$$y_{pr_{k-1}(p)} = 1 - v_p^{\text{tr}}, \quad \forall p \in O, \tag{7.3}$$

where, for $k = 1$, $r_{k-1}(p)$ is the previous room $r_{\text{prev}}(p)$ patient $p \in O$ was assigned to. This constraint adds a penalty term if occupant p is not assigned to that room. These variables are added to the objective function with weight β_{tr}. For interval $k = 2, \ldots, K$, the constraint can remain as is, forcing patients to remain in the same room as in the previous interval. Making these changes ensures that only current occupants can be transferred. The updated ILP is given in Formulation (A.13). Note that we use here that $P^{k-1} \cap P^k = O$ for $k = 1$.

To account for the previously assigned nurses $\mathcal{N}_{\text{prev}}(p)$ of a patient $p \in O$, we do not have to change the ILP. In the offline algorithm, a set $\mathcal{N}_k(p)$ was defined for each interval $k = 1, \ldots, K$ and each patient $p \in P$. This set denoted the set of nurses that was assigned to patient p before interval k. For $k = 1$, this set was empty. After each iteration, the set was updated based on the nurse-to-patient assignment. To account for $\mathcal{N}_{\text{prev}}(p)$, we can simply initialize $\mathcal{N}_k(p) = \mathcal{N}_{\text{prev}}(p)$ for $k = 1$ and $p \in O$. Together with Constraint (5.12), this ensures that for $p \in O$ a nurse $n \in \mathcal{N}_{\text{prev}}(p)$ already counts towards the continuity of care objective.

Lastly, similar to the greedy heuristic, we can use the schedule made on a previous day. In this case however, we can set both variables y_{pr} and x_{nrs}. In particular, for $p \in O \cup P_{\text{old}}$ and $r \in R$, we initialize the value of y_{pr} using the previous schedule. Similarly, for $s \in S_{\text{prev}}$, $n \in N(s)$ and $r \in R$, we initialize the value of x_{nrs} using the previous schedule.

7.4 Online simulated annealing

The simulated annealing algorithm described in Section 5.4 can be adapted to an online heuristic by making several changes.

To allow the current occupants to transfer rooms, we must make some changes to how the MovePatient and SwapPatients moves are randomly selected. In particular, we must now allow all patients $p \in P$ to change rooms, instead of only the non-occupants.

To randomly select a MovePatient move, we first randomly select a patient $p \in P$ with $|R(p)| \geq 2$. Afterwards, we randomly select a new room $r \in R(p) \setminus \{r_k(p)\}$, where $r_k(p)$ denotes the assigned room of patient p during in solution Q_k.

To randomly select a SwapPatients move, we randomly select a pair of patients (p_1, p_2) from the set

$$\begin{aligned} \{(p_1, p_2) : {} & p_1, p_2 \in P,\ p_1 \neq p_2, \\ & r_k(p_1) \in R(p_2),\ r_k(p_2) \in R(p_1), \\ & r_k(p_1) \neq r(p_2)\}. \end{aligned} \tag{7.4}$$

For both the MovePatient and SwapPatients moves, we must also make a change to how the change in objective Δ_k is calculated. In particular, if an occupant $p \in O$ is moved from the room they were assigned to originally, i.e., from room $r_{\text{prev}}(p)$, then a penalty with weight β_{tr} is added. If the occupant is moved back to that room, the same weight is subtracted from Δ_k. This ensures that transfers are correctly taken into account.

In addition to changes to the MovePatient and SwapPatients moves, we also make a change to the RemoveNurse move. Because the nurse assignment in previous shifts is already fixed, some nurses cannot be removed from the nurse set $\mathcal{N}_k(p)$ for a patient $p \in O$. In particular, these are nurses that are also in $n \in \mathcal{N}_{\text{prev}}(p)$. This can limit the number of RemoveNurse moves that are available.

Lastly, we can use the previous schedule as an initial solution Q_0 in the simulated annealing algorithm. In particular, we construct an initial solution in the following way:

- Patients $p \in O \cup P_{\text{old}}$ were included in a previous schedule and so we can assign them to the room that they were assigned to.
- Patients $p \in P_{\text{new}}$ are newly registered patients and were therefore not assigned to a room in the previous schedule. We assign them randomly to a room $r \in R(p)$, i.e., a room that is compatible for patient p. This is similar to how the initial solution was determined in the offline algorithm.
- The shifts $s \in S_{\text{old}}$ were considered in the previous schedule, so we can use the nurse-to-room assignment of those shifts for the initial solution.
- The shifts $s \in S \setminus S_{\text{old}}$ were not considered in the previous schedule. For those shifts, we must do a random assignment of nurses. Similar to the offline algorithm, this is done by, for each room $r \in R$ and $s \in S \setminus S_{\text{old}}$, randomly assigning a nurse $n \in N(s)$

This gives an initial solution that can be used in simulated annealing.

7.5 Online evaluation

In this section, we compare the online versions of each of the heuristics. For each heuristic, the same parameters and design decisions as described in Chapter 5 are used. In Section 7.5.1, we describe the data used to evaluate these heuristics and how the evaluation is performed. Then, in Section 7.5.2, the results are shown. Furthermore, in Section 7.5.3, the impact of the transfer weight β_{tr} is determined.

7.5.1 Data and experiment setup

In order for us to evaluate the results in an online setting, we transform the datasets used throughout this thesis into online variants. This allows us to compare the results from the online and offline settings. To transform the data, we must select several patients and label them as emergency patients. In particular, each (non-occupant) patient has a probability π of being selected as an emergency patient. In reality, the value for π depends on, for example, the region, department and time of the year. Several values are therefore tested. Based on data from Dutch academic

hospital Erasmus MC, emergency patients make up roughly 11.6% of patients in a ward. For this reason, we use 0.1 and 0.15 as values for π. For each emergency patient, we must also randomly generate the day at which they become known to the planner, known as the registration day d_{reg}. We assume that this is either 0, 1 or 2 days before their arrival. This value is selected randomly with probability 0.4, 0.4 and 0.2, respectively.

For each original instance and each value of π, five new online instances are randomly generated. Note that, if a newly generated online instance contained no emergency patients, then we keep resampling until there is.

Now that we have a dataset on which to evaluate the heuristics, we must determine how this is done exactly. Let D_{full} denote the full planning horizon on which we are making and evaluating the schedule. This set is the same as the set D used in the offline evaluation. Based on the registration days of the emergency patients, we have several days on which we must update the schedule. These are denoted using $D_{\text{u}} \subseteq D_{\text{full}}$. For each schedule update day $d_{\text{u}} \in D_{\text{u}}$, we consider the planning horizon until the end of the full planning horizon. If we denote this with $D_{d_{\text{u}}}$ for $d_{\text{u}} \in D_{\text{u}}$, then

$$D_{d_{\text{u}}} = \{d_{\text{u}}, d_{\text{u}} + 1, \ldots, |D_{\text{full}}|\}\,. \tag{7.5}$$

Each day d_{u}, the previous schedule must be updated to fit in the emergency patients. However, the first day of the full planning horizon D_{full} does not have a previous schedule. In that case, the assignments must be made from the beginning. This means that we can not use the warm start for the greedy and interval heuristics. For the simulated annealing algorithm, we can not use the previous schedule for the initial solution and must therefore use the offline initialization procedure discussed in Section 5.4.1.1, but only taking into account the currently registered patients. Also note that for each occupant p of the first day, the set $\mathcal{N}_{\text{prev}}(p) = \emptyset$.

Recall that in Chapter 6, the heuristics were evaluated using the same total runtime limit. This runtime limit was defined by looking at the total length of the planning horizon. For each week, 2.5 minutes of runtime was given. In the online setting, this must slightly change. Each time the schedule is updated, the length of the planning horizon decreases and might no longer be multiple weeks. Instead of allocating 2.5 minutes each week, we instead give 20 seconds for each day. This means that the runtime limit is slightly lower than it was originally. For example, a planning horizon of four weeks was originally given 10 minutes and is now given 9 minutes and 20 seconds.

Despite of these lower runtime limits, the time it takes to evaluate one algorithm for a single instance becomes much larger. Consider for example instance m01. In the offline setting, this instance is given a total runtime limit of 10 minutes. In the online setting, the schedule is revised multiple times, which adds additional runtime. When looking at the five online instances of m01 with emergency probability $\pi = 0.1$, the average maximum runtime is 41 minutes and 24 seconds. This means it takes much longer to evaluate the heuristics in an online setting. This is an even bigger problem if the value of π is increased or a larger instance is considered, since the schedule is then revised more often. For this reason, we only run the evaluation for the small instances.

In addition to running the heuristics only for the smaller instances, we also change how often we run each heuristic. Recall that throughout this thesis, we ran each heuristic ten times for each instance to reduce randomness in the heuristics.

When evaluating the online heuristic, we create 5 online instances per value of $\pi \in \{0.1, 0.15\}$ for each instance. We run each heuristic only once per online instance. This means that, in total, we still run a heuristic ten times per instance, but these runs are divided across the different online instances and the different values of π.

One problem of the way D_{d_u} is defined for each $d_\mathrm{u} \in D_\mathrm{u}$ is related to the interval heuristic and how the time budget is divided between intervals. Since planning horizons in the offline setting were always a certain number of weeks, there was never a large difference in interval lengths. In the online setting however, there might be a large difference. For example, we could have a planning horizon of planning horizon of $|D_{d_\mathrm{u}}| = 8$ days. For a small instance with $L_{\max} = 7$, this gives two intervals of length 7 and 1. The problem arises when we assign the runtime budget to each interval. Originally, we divided the total runtime budget evenly across the intervals. However, in this example, the interval with length 7 is much harder to solve than the one with length 1. For this reason, we divide the runtime based on the lengths of the intervals. In this example, the interval of a single day will only receive 1/8th of the total runtime budget.

We must also make a small change to how the simulated annealing algorithm is run. In the offline setting, we first run the algorithm for $K = 1{,}000{,}000$ to determine roughly how many iterations are needed to reach the runtime budget. In the online setting, this is not only done once, but done every time the schedule is updated.

7.5.2 Results

Now that we have discussed how the heuristics are adapted to an online setting and how they are evaluated, we can run the experiments. As stated previously, we only run the heuristic for the small instances because of the time it takes to evaluate them. In this section, we assume that the transfer weight is $\beta_{\mathrm{tr}} = 1$. This choice is examined in Section 7.5.3. In Table 7.1, the results for running all heuristics are shown. Note that we also show the results for the offline heuristics, i.e., when $\pi = 0$. These were taken from Tables 6.1-6.3.

Table 7.1: Objective values for different heuristics and different emergency probabilities π.

Instance	π	Greedy (seq)	Greedy (non-seq)	Interval heuristic	Simulated annealing
m01	0	906.882	872.033	824.908	835.427
m01	0.1	899.817	875.547	822.677	833.273
m01	0.15	897.323	874.853	815.830	834.103
m02	0	323.045	279.950	274.118	277.552
m02	0.1	322.847	280.897	275.123	277.443
m02	0.15	318.717	281.673	277.297	281.060
m03	0	289.900	277.183	258.433	260.672
m03	0.1	288.210	278.660	260.027	263.487
m03	0.15	287.993	279.167	260.957	263.323
m04	0	471.347	434.700	429.198	425.912
m04	0.1	476.913	441.147	445.587	436.763
m04	0.15	465.467	444.627	455.470	441.927
m05	0	499.373	455.283	450.187	437.525
m05	0.1	494.503	458.487	449.423	447.947
m05	0.15	492.943	459.350	446.937	452.723

Perhaps most interestingly, we see no large difference between the results from the offline and the online settings. Because the heuristics in the online settings work with incomplete information, we would expect the final objective value to be higher. Furthermore, we would expect this to worsen as the value of π increases. However, we see that this is not necessarily the case. In fact, for some instances and heuristics, the found solution has a lower objective value than the offline solution. There could be several explanations for this.

One reason why this could happen is because we now allow transfers. By allowing transfers for the occupants each time the schedule is updated, the solution space becomes larger. Because of this, the optimal solution in the online setting could have a smaller objective value than in the offline setting. This also means that the heuristics also might get a lower objective value.

In order to test this, we can run the same experiment, but with transfer weight $\beta_{\text{tr}} = 5$. Note that for the two greedy heuristics, all found solutions contained no transfers. That is because those heuristics were designed to only allow transfers if there was no other solution. Changing the weight β_{tr} does nothing for those two heuristics as this weight is not taken into account. For the other two heuristics, changing the weight does impact the solution. In Table 7.2, the results from running the interval heuristic again using $\beta_{\text{tr}} = 5$ are shown. The other objective component weights are still set to 1. Again we see no large difference. Even when β_{tr} is increased and the number of transfers is reduced, the solution still has a similar objective value as the offline solution.

Table 7.2: Average objective value and number of transfers for running the interval heuristic with $\beta_{\text{tr}} = 1$ and $\beta_{\text{tr}} = 5$. Results are averaged over 5 runs. Note that the objective value presented includes the weighted number of transfers based on β_{tr}. Shows results for the interval heuristic and for simulated annealing.

Interval heuristic					
Instance	π	$\beta_{\text{tr}} = 1$		$\beta_{\text{tr}} = 5$	
		Objective	Transfers (#)	Objective	Transfers (#)
m01	0.1	822.677	8.6	820.590	0
m01	0.15	815.830	7	820.257	0
m02	0.1	275.123	0	274.857	0
m02	0.15	277.297	0.2	275.980	0
m03	0.1	260.027	1	260.313	0
m03	0.15	260.957	1.4	260.363	0
m04	0.1	445.587	9.2	430.073	0.8
m04	0.15	455.470	16.6	448.240	3.4
m05	0.1	449.423	4	445.147	0.2
m05	0.15	446.937	5	446.903	0.6

Simulated annealing					
Instance	π	$\beta_{\text{tr}} = 1$		$\beta_{\text{tr}} = 5$	
		Objective	Transfers (#)	Objective	Transfers (#)
m01	0.1	833.273	12	836.077	1.4
m01	0.15	834.103	11.8	825.773	0.6
m02	0.1	277.443	0.2	279.183	0
m02	0.15	281.060	1	280.097	0
m03	0.1	263.487	2.6	262.467	0
m03	0.15	263.323	1.8	263.173	0
m04	0.1	436.763	13.4	435.940	1.2
m04	0.15	441.927	23	427.820	1
m05	0.1	447.947	9.6	441.723	0.2
m05	0.15	452.723	17.4	441.520	0

Another possible explanation for why the objective value for the online heuristic is very similar to the offline heuristics, is the time spent on a single instance. As was stated before, the total maximum runtime of an online instance could be much larger than for an offline instance. The maximum runtime for creating the initial schedule was roughly the same as in the offline case, but each time the schedule is revised, the heuristic spends additional time finding a solution. It could be that giving additional time allows the online heuristics to find better quality solutions, which compensates for the incomplete information of the online setting.

To see whether this truly is the case, we can run the evaluation from Chapter 6, but use the larger runtimes. In particular, for each instance, we use the average maximum runtime of the ten corresponding online instances. For the simulated annealing algorithm, we first perform a single run with $K = 10{,}000{,}000$ and then use the procedure described in Section 6.1 to determine a value for K that has approximately the desired runtime. This initial value of K is larger than the one used in Section 6.1 since we now consider a much larger maximum runtime.

The results can be found in Table 7.3. Note that the sequential greedy heuristic was not included because the additional runtime has no effect on the solution. We

see that, although the increased runtime improves the obtained objective values, it is still very similar to the values found in Table 7.1.

Table 7.3: Offline heuristic results with increased runtimes.

Instance	Runtime limit (s)	Greedy (non-seq)	Interval heuristic	Simulated annealing
m01	2594	872.033	821.457	828.450
m02	1758	279.950	273.890	276.067
m03	760	277.183	258.433	261.525
m04	1104	434.283	416.810	420.073
m05	882	454.900	446.033	435.083

Because the similar objective values are not necessarily the result of adding transfers or allowing extra runtime, we conclude that the heuristics generally work well in an online setting, at least for small instances. Whether this is also true for larger instances is left for future research.

7.5.3 Transfer weight

During the evaluation, we set the transfer weight $\beta_{\text{tr}} = 1$. In this section, we vary this weight to see the impact it has on the other objective components. To do this, we use a similar approach as in Section 6.2, where this was done for all objective components. As we now consider six different objective components, we scale the objectives such that

$$\beta_{\text{cc}} + \beta_{\text{gm}} + \beta_{\text{sv}} + \beta_{\text{wv}} + \beta_{\text{wi}} + \beta_{\text{tr}} = 6. \tag{7.6}$$

To test the impact of the transfer weight β_{tr}, we try various values. In particular, we test values $\beta_{\text{tr}} \in \{0, 1, \ldots, 5, 6\}$. The other weights are then set to the same values such that Equation (7.6) is satisfied. This is the same approach that was used in Section 6.2. Note that this is different from what was done in Table 7.2, where only the value β_{tr} changed while the other weights remained at 1.

We test the impact of transfer weight β_{tr} on instance m01. In particular, we run the interval heuristic for all ten online instances corresponding to instance m01. Since the results for online instances with $\pi = 0.1$ and online instances with $\pi = 0.15$ were very similar, the results are combined. We use the same runtime and parameters as described in Section 7.5.1.

The impact of the transfer weight is shown in Figure 7.1. Note that component score was defined in Section 6.2 and that we use the same lower and upper bounds for the objective components. For the transfer objective, we use a lower bound of 0. The upper bound is calculated by summing the number of occupants each time the schedule is updated. Since this can be different for each online instance, the upper bound used is the average of this value across the ten different online instances. This gave an upper bound of 46.3.

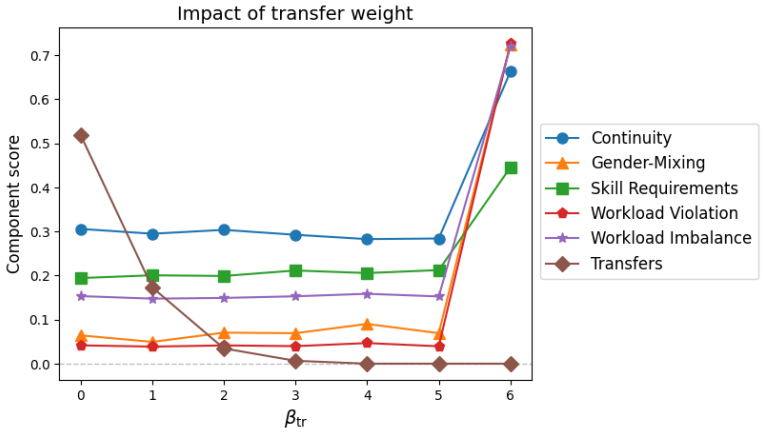


Figure 7.1: Results of sensitivity analysis. Each subplot shows how each objective component changes when a singular objective weight is changed.

We can clearly see that the number of transfers decreases as the transfer weight β_{tr} increases. This is as expected since the number of transfers is more important in the objective function. Apart from that, the values for the other objective components do not change much. Even when $\beta_{\text{tr}} = 0$, there is little change. This suggests that allowing transfers can only slightly improve the other objective components. As transfers are very disruptive, it makes sense to increase the transfer weight. This does not worsen the other objective components significantly.

Chapter 8

Conclusion and discussion

This thesis aimed to solve the PNRA problem while focusing on the continuity of care. Since improved continuity of care leads to better patient satisfaction and health outcomes, this is important to take into account during the assignment process. In the PNRA problem, this is done by counting the number of different nurses that are assigned to a patient throughout their stay in the hospital.

The PNRA problem can be modeled using an ILP formulation. However, the resulting ILP can only be solved within a reasonable time for very small instances. For this reason, heuristics are needed to find high-quality solutions quickly. In this thesis, four different heuristics were developed:

- Greedy sequential heuristic: a greedy heuristic that solves the problem in two phases. In the first phase, patients are sequentially assigned to rooms such that patients are roughly evenly divided across the rooms. It also uses a theoretical nurse assignment that minimizes continuity of care to place patients with similar nurses in the same room. In the second phase, nurses are assigned to rooms using an ILP that considers each shift sequentially, using the nurse assignment in previous shifts to optimize the continuity of care value.
- Greedy non-sequential heuristic: a greedy heuristic that also solves the problem in two phases. The first phase is the same as the sequential variant. The second phase uses an ILP to determine the nurse assignment for all shifts simultaneously.
- Interval heuristic: a heuristic that solves the problem by dividing the planning horizon into several intervals. Each interval is solved sequentially and uses the nurse assignments from previous intervals to optimize the continuity of care value.
- Simulated annealing: a local search metaheuristic that uses four different move types to traverse the search space. The moves either change the room of a patient, swap the rooms of two patients, change the nurse assigned to a room or remove a nurse from assignments for a singular patient. This last move aims to reduce the continuity of care value for a single patient by changing the nurse assignment each time a specific nurse is assigned.

Generally, the heuristics perform very well. Sometimes, they were even able to find a better solution in a few minutes than the ILP found in 8 hours. The performance of the heuristics depends on the size of the problem, determined by the number

of patients. For smaller instances, the best solutions were found using the interval heuristic and simulated annealing. The two greedy heuristics performed worse, mostly because of the patient-to-room assignment phase.

For medium and large instances, the interval heuristic performs much worse due to the increased size of the ILPs and the fact that we consider shorter intervals. In those cases, the greedy non-sequential heuristic and simulated annealing perform better. For the medium instances, both heuristics found solutions with similar objective values. For the large instances, the greedy non-sequential heuristic almost always found the best solutions.

The four heuristic were additionally tested in an online setting, in which patients could unexpectedly arrive as emergency patients. Since instances could become infeasible due to previous decisions, transfers of current occupants were allowed to ensure feasibility. Only minimal changes to the heuristics were needed to facilitate this. Due to the increased computation time needed to solve these online settings, only the small instances were evaluated. It was found that the performance of the heuristics in the online settings is very similar to the offline setting. Similar to the offline setting, the interval heuristic and simulated annealing heuristic work best.

There are several avenues for future research. Firstly, the data used throughout this thesis was created using an instance generator that was not based on real life data. This means that possible correlations in patient workload, skill requirements and length of stay were not taken into account when generating the data. This means that the performance of these heuristics might not translate well to real life data. A future study could investigate whether this is the case.

Additionally, the method for dividing the instances into different sizes could be changed. Throughout this thesis, the various instances were divided into small, medium and large instances based on the number of patients. However, not every instance behaved similarly to other instances of the same size. For example, the interval heuristic for instance m07 was much faster compared to other medium instances, which is more similar to the small instances. One possible way to fix this is by dividing the instances into even more size categories. This, however, means that there are less instances of a particular size, which increases the chance of overfitting. Another possible fix is by using other criteria to divide the instances. More research is needed to find an appropriate method to do so.

Furthermore, the comparison of instances of the same size could be improved. Recall that objective score was computed by scaling the objective value to be between 0 and 1 using lower and upper bounds. During the tuning procedure, parameters were chosen that attained the lowest average score across all instances of a specific size. However, because of this method, the objective value for instances with tighter bounds is weighted more in the average score. This means that instances that are larger or more difficult are weighted less since it is more difficult to obtain good bounds. The impact of this must be investigated in the future.

Although the heuristics presented in this thesis perform quite well, it might be possible to improve the performance in various ways. This could be done by considering different heuristic design options such as different sorting methods for the greedy heuristics, different splitting methods for the interval heuristic or introducing additional moves in simulated annealing. Additionally, different new heuristics could be implemented that might yield better results. Whether such changes actually lead to improved solutions is left for further research.

Bibliography

Abera, A. K., O'Reilly, M. M., Fackrell, M., Holland, B. R., & Heydar, M. (2020). On the decision support model for the patient admission scheduling problem with random arrivals and departures: A solution approach. *Stochastic Models*, *36*(2), 312–336. https://doi.org/10.1080/15326349.2020.1742161

Abu Doush, I., Al-Betar, M. A., Awadallah, M. A., Hammouri, A. I., Al-Khatib, R. M., Elmustafa, S., & Alkhraisat, H. (2020). Harmony Search Algorithm for Patient Admission Scheduling Problem. *Journal of Intelligent Systems*, *29*(1), 540–553. https://doi.org/10.1515/jisys-2018-0094

ACCESS NL. (2018). Visiting dutch hospitals [accessed April 17, 2026]. https://access-nl.org/healthcare-netherlands/dutch-healthcare-system/hospitals/accommodation-and-facilities-dutch-hospitals/#questions-1740

Allen, S. B. (2015). The nurse-patient assignment: Purposes and decision factors. *JONA: The Journal of Nursing Administration*, *45*(12), 628–635. https://doi.org/10.1097/nna.0000000000000276

Baker, R. L., Tindell, S., Deborah, R.-B., Behan, B., Turpin, P. G., Rosenberger, J. M., & Punnakitikashem, P. (2010). Phase I Creating an Electronic Prototype to Generate Equitable Hospital Nurse-to-Patient Assignments. *CIN - Computers Informatics Nursing*, *28*(1), 57–62. https://doi.org/10.1097/ncn.0b013e3181c0472c

Bolaji, A. L. a., Bamigbola, A. F., & Shola, P. B. (2018). Late acceptance hill climbing algorithm for solving patient admission scheduling problem. *Knowledge-Based Systems*, *145*, 197–206. https://doi.org/10.1016/j.knosys.2018.01.017

Borchani, R., Masmoudi, M., Jarboui, B., & Siarry, P. (2021). Heuristics-based on the Hungarian Method for the Patient Admission Scheduling Problem. In *Operations research and simulation in healthcare* (pp. 33–62). https://doi.org/10.1007/978-3-030-45223-0_2

Bortoletto, L., Da Silva, R. F. F., Da Silva, J. P. F., Marão, A., Romero, I., Saraiva, R. V., & Schouery, R. C. S. (2025). Multi-Stage Integer Linear Programming Framework for Integrated Healthcare Planning. *Operations Research Data Analytics and Logistics*, *47*, 200506. https://doi.org/10.1016/j.ordal.2026.200506

Brandt, T., Büsing, C., & Engelhardt, F. (2025). Patient-to-room assignment with single-rooms entitlements: Combinatorial insights and integer programming formulations. *European Journal of Operational Research*, *325*(1), 20–37. https://doi.org/10.1016/j.ejor.2025.02.018

Brandt, T., Büsing, C., & Knust, S. (2024). Structural insights about avoiding transfers in the patient-to-room assignment problem. *Discrete Applied Mathematics*, *347*, 231–248. https://doi.org/10.1016/j.dam.2023.12.025

Brandt, T., Klein, T. L., Reuter-Oppermann, M., Schäfer, F., Thielen, C., de Vrugt, M. v., & Viana, J. (2025). Integrated patient-to-room and nurse-to-patient

assignment in hospital wards. *OR Spectrum*, 1–44. https://doi.org/10.1007/s00291-024-00800-z

Carello, G., Galperti, L., & Lanzarone, E. (2024). Realization-based robust assignments for a nurse-to-patient assignment problem in home health care services. *Flexible Services and Manufacturing Journal*, *38*(1), 392–440. https://doi.org/10.1007/s10696-024-09577-3

Carello, G., & Lanzarone, E. (2014). A cardinality-constrained robust model for the assignment problem in home care services. *European Journal of Operational Research*, *236*(2), 748–762. https://doi.org/https://doi.org/10.1016/j.ejor.2014.01.009

Ceschia, S., Rosati, R. M., Schaerf, A., Smet, P., Vanden Berghe, G., & Zanazzo, E. (2024). IHTC 2024 [accessed December 3rd, 2025]. https://ihtc2024.github.io/

Ceschia, S., Rosati, R. M., Schaerf, A., Smet, P., Vanden Berghe, G., & Zanazzo, E. (2025). The integrated healthcare timetabling competition 2024. *Operations Research, Data Analytics and Logistics*, *45*, 200481. https://doi.org/10.1016/j.ordal.2025.200481

Ceschia, S., & Schaerf, A. (2011). Local search and lower bounds for the patient admission scheduling problem. *Computers and Operations Research*, *38*(10), 1452–1463. https://doi.org/10.1016/j.cor.2011.01.007

Ceschia, S., & Schaerf, A. (2012). Modeling and solving the dynamic patient admission scheduling problem under uncertainty. *Artificial Intelligence in Medicine*, *56*(3), 199–205. https://doi.org/10.1016/j.artmed.2012.09.001

Ciccarelli, F., Di Biase, A., & Furini, F. (2025). A Two-Phase Matheuristic for the Integrated Healthcare Timetabling Problem. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.5768118

Davison, M., Burgess, G., Hamm, R., Lowery, B., & Page, A. (2025). A parallelised hyper-heuristic framework for the Integrated Healthcare Timetabling Competition 2024. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.5601273

De Maeseneer, J. M., De Prins, L., Gosset, C., & Heyerick, J. (2003). Provider continuity in family medicine: Does it make a difference for total health care costs? *The Annals of Family Medicine*, *1*(3), 144–148. https://doi.org/10.1370/afm.75

Demeester, P., Souffriau, W., Causmaecker, P. D., & Berghe, G. V. (2010). A hybrid tabu search algorithm for automatically assigning patients to beds. *Artificial Intelligence in Medicine*, *48*(1), 61–70. https://doi.org/10.1016/j.artmed.2009.09.001

DHPC. (2024). DelftBlue Supercomputer (Phase 2). *Delft High Performance Computing Centre*. https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2

Farzanegan, F., Sepehri, M. M., Samani, M. R. G., & Barati, L. (2025). Integrating artificial intelligence and optimization tools in bed assignment to enhance patient satisfaction under uncertainty. *Engineering Applications of Artificial Intelligence*, *159*, 111363. https://doi.org/10.1016/j.engappai.2025.111363

Fisher, M. L., Jaikumar, R., & Van Wassenhove, L. N. (1986). A multiplier adjustment method for the generalized assignment problem. *Management science*, *32*(9), 1095–1103. https://doi.org/10.1287/mnsc.32.9.1095

Garey, M., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman & Co.

Gray, D. J., Sidaway-Lee, K., White, E., Thorne, A., & Evans, P. H. (2018). Continuity of care with doctors - A matter of life and death? A systematic review of continuity of care and mortality. *BMJ Open*, *8*(6), e021161. https://doi.org/10.1136/bmjopen-2017-021161

Guericke, D., van der Hulst, R., Karimpour, A., Schrader, I., & Walter, M. (2025). A hybrid solution approach for the Integrated Healthcare Timetabling Competition 2024. *Operations Research, Data Analytics and Logistics*, *46*, 200502. https://doi.org/10.1016/j.ordal.2026.200502

Guido, R. (2024). Patient admission scheduling problems with uncertain length of stay: Optimization models and an efficient matheuristic approach. *International Transactions in Operational Research*, *31*(1), 53–87. https://doi.org/10.1111/itor.13272

Gulliford, M., Naithani, S., & Morgan Myfanwy. (2006). What is 'continuity of care'? *Journal of Health Services Research & Policy*, *11*(4), 248–250. https://doi.org/10.1258/135581906778476490

Gurobi Optimization, LLC. (2026). Gurobi Optimizer Reference Manual. https://www.gurobi.com

Güven-Koçak, Ş., Heching, A., Keskinocak, P., & Toriello, A. (2024). Continuity of care in home health care scheduling: A rolling horizon approach. *Journal of Scheduling*, *27*(4), 375–392. https://doi.org/10.1007/s10951-023-00796-4

Hammouri, A. I. (2022). A modified biogeography-based optimization algorithm with guided bed selection mechanism for patient admission scheduling problems. *Journal of King Saud University - Computer and Information Sciences*, *34*(3), 871–879. https://doi.org/10.1016/j.jksuci.2020.01.013

Karp, R. M. (1972). Reducibility among Combinatorial Problems. In *Complexity of computer computations* (pp. 85–103). https://doi.org/10.1007/978-1-4684-2001-2_9

Kifah, S., & Abdullah, S. (2015). An adaptive non-linear great deluge algorithm for the patient-admission problem. *Information Sciences*, *295*, 573–585. https://doi.org/10.1016/j.ins.2014.10.004

Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Kratz, M., Pfau, J., Zschaler, S., Kosiol, J., & Schürr, A. (2025). High-Level Specification of MILP Problems with Class Diagrams and Graph Transformations: A Case Study in Model-Driven Optimisation of the Integrated Healthcare Timetabling Problem. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.5768119

Ku, W.-Y., Pinheiro, T., & Beck, J. C. (2014). CIP and MIQP Models for the Load Balancing Nurse-to-Patient Assignment Problem. *Principles and Practice of Constraint Programming*, 424–439. https://doi.org/10.1007/978-3-319-10428-7_32

Lanzarone, E., Matta, A., & Sahin, E. (2012). Operations management applied to home care services: The problem of assigning human resources to patients. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *42*(6), 1346–1363. https://doi.org/10.1109/TSMCA.2012.2210207

Liu, H., Wang, Y., & Hao, J.-K. (2024). Solving the patient admission scheduling problem using constraint aggregation. *European Journal of Operational Research*, *316*(1), 85–99. https://doi.org/10.1016/j.ejor.2024.02.009

Lusby, R. M., Schwierz, M., Range, T. M., & Larsen, J. (2016). An adaptive large neighborhood search procedure applied to the dynamic patient admission scheduling problem. *Artificial Intelligence in Medicine*, *74*, 21–31. https://doi.org/10.1016/j.artmed.2016.10.002

Marzouk, M., & Kamoun, H. (2021). Nurse to patient assignment through an analogy with the bin packing problem: Case of a Tunisian hospital. *Journal of the Operational Research Society*, *72*(8), 1808–1821. https://doi.org/10.1080/01605682.2020.1727300

Mullinax, C., & Lawley, M. (2002). Assigning patients to nurses in neonatal intensive care. *Journal of the Operational Research Society*, *53*(1), 25–35. https://doi.org/10.1057/palgrave.jors.2601265

National Health Service. (2019). Delivering same-sex accommodation [accessed April 17, 2026]. https://www.england.nhs.uk/long-read/delivering-same-sex-accommodation/

Nguyen, N.-D., Lahrichi, N., & Yu, C. (2025). Nurse-to-patient assignment problem with uncertainties in demand and skill requirements: A stochastic programming approach. *Computers & Industrial Engineering*, *210*, 111554. https://doi.org/10.1016/j.cie.2025.111554

NSW Government. (2022). Same gender accommodation [accessed April 17, 2026]. https://www1.health.nsw.gov.au/pds/Pages/doc.aspx?dn=PD2022_042

O'Reilly, M. M., Krasnicki, S., Montgomery, J., Heydar, M., Turner, R., Dam, P. V., & Maree, P. (2025). Markov decision process and approximate dynamic programming for a patient assignment scheduling problem. *Annals of Operations Research*, *347*(3), 1493–1531. https://doi.org/10.1007/s10479-025-06553-4

Punnakitikashem, P., Rosenberber, J. M., Buckley Behan, D., & Goss, K. (2006). An Optimization-Based Prototype for Nurse Assignment. *Proceedings of the 7th Asian Pacific Industrial Engineering and Management Systems Conference*.

Punnakitikashem, P., Rosenberber, J. M., & Buckley-Behan, D. F. (2013). A stochastic programming approach for integrated nurse staffing and assignment. *IIE Transactions*, *45*(10), 1059–1076. https://doi.org/10.1080/0740817X.2012.763002

Range, T. M., Lusby, R. M., & Larsen, J. (2014). A column generation approach for solving the patient admission scheduling problem. *European Journal of Operational Research*, *235*(1), 252–264. https://doi.org/10.1016/j.ejor.2013.10.050

Rappos, E., Varani, J., & Robert-Nicoud, S. (2025). A MIP approach for solving the integrated healthcare timetable problem. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.5768120

Schäfer, F., Walther, M., Grimm, D. G., & Hübner, A. (2023). Combining machine learning and optimization for the operational patient-bed assignment problem. *Health Care Management Science*, *26*(4), 785–806. https://doi.org/10.1007/s10729-023-09652-5

Schaus, P., & Régin, J.-C. (2014). Bound-consistent spread constraint: Application to load balancing in nurse-to-patient assignments. *EURO Journal on Computational Optimization*, *2*(3), 123–146. https://doi.org/10.1007/s13675-013-0018-8

Schaus, P., Van Hentenryck, P., & Régin, J.-C. (2009). Scalable Load Balancing in Nurse to Patient Assignment Problems. *Integration of AI and OR Tech-*

niques in Constraint Programming for Combinatorial Optimization Problems, 248–262. https://doi.org/10.1007/978-3-642-01929-6_19

Schmidt, R., Geisler, S., & Spreckelsen, C. (2013). Decision support for hospital bed management using adaptable individual length of stay estimations and shared resources. *BMC Medical Informatics and Decision Making*, *13*, 3. https://doi.org/10.1186/1472-6947-13-3

Turhan, A. M., & Bilgen, B. (2017). Mixed integer programming based heuristics for the Patient Admission Scheduling problem. *Computers & Operations Research*, *80*, 38–49. https://doi.org/10.1016/j.cor.2016.11.016

Van Oostveen, C. J., Braaksma, A., & Vermeulen, H. (2014). Developing and Testing a Computerized Decision Support System for Nurse-to-Patient Assignment. *CIN - Computers Informatics Nursing*, *32*(6), 276–285. https://doi.org/10.1097/CIN.0000000000000056

Van Walraven, C., Oake, N., Jennings, A., & Forster, A. J. (2010). The association between continuity of care and outcomes: A systematic and critical review. *Journal of Evaluation in Clinical Practice*, *16*(5), 947–956. https://doi.org/10.1111/j.1365-2753.2009.01235.x

Vancroonenburg, W., Causmaecker, P. D., & Berghe, G. V. (2016). A study of decision support models for online patient-to-room assignment planning. *Annals of Operations Research*, *239*(1), 253–271. https://doi.org/10.1007/s10479-013-1478-1

Vancroonenburg, W., Della Croce, F., Goossens, D., & Spieksma, F. C. (2014). The Red-Blue transportation problem. *European Journal of Operational Research*, *237*(3), 814–823. https://doi.org/10.1016/j.ejor.2014.02.055

Zanazzo, E., Ceschia, S., & Schaerf, A. (2025). Multi-neighborhood simulated annealing for the integrated patient-to-room and nurse-to-patient assignment problem. *Flexible Services and Manufacturing Journal*. https://doi.org/10.1007/s10696-025-09591-z

Zanazzo, E., Smet, P., Ceschia, S., Rosati, R. M., Schaerf, A., & Berghe, V. (2025). A Multi-Neighborhood Simulated Annealing Approach to the Integrated Healthcare Timetabling Problem. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.5376065

Appendix A

Integer linear programs

A.1 Lower bounds

A.1.1 Workload violation (partial)

$$
\begin{aligned}
\min \quad & \sum_{n \in N} \sum_{s \in S(n)} v_{ns}^{\mathrm{wv}} \\
\text{s.t.} \quad & \sum_{n \in N(s)} z_{nps} = 1, && \forall\, p \in P,\, s \in S(p), \\
& \sum_{p \in P(s)} w(p,s) \cdot z_{nps} \leq w_{\max}(n,s) + v_{ns}^{\mathrm{wv}}, && \forall\, n \in N,\, s \in S(n), \\
& z_{nps} \in \{0,1\}, && \forall\, n \in N,\, p \in P,\, s \in S(n,p), \\
& v_{ns}^{\mathrm{wv}} \geq 0, && \forall\, n \in N,\, s \in S(n).
\end{aligned}
\tag{A.1}
$$

A.1.2 Workload imbalance (partial)

$$
\begin{aligned}
\min \quad & \sum_{s \in S} l_s^+ - l_s^- \\
\text{s.t.} \quad & \sum_{n \in N(s)} z_{nps} = 1, && \forall\, p \in P,\, s \in S(p), \\
& l_s^- \leq \sum_{p \in P(s)} \frac{w(p,s)}{w_{\max}(n,s)} \cdot z_{nps} && \forall\, n \in N,\, s \in S(n), \\
& l_s^+ \geq \sum_{p \in P(s)} \frac{w(p,s)}{w_{\max}(n,s)} \cdot z_{nps} && \forall\, n \in N,\, s \in S(n), \\
& z_{nps} \in \{0,1\}, && \forall\, n \in N,\, p \in P, s \in S(n,p), \\
& l_s^-,\, l_s^+ \geq 0, && \forall\, s \in S,
\end{aligned}
\tag{A.2}
$$

A.1.3 Gender-mixing (partial)

$$
\begin{aligned}
\min \quad & \sum_{r\in R}\sum_{s\in S_{\text{early}}} v^{\text{gm}}_{rs} \\
\text{s.t.} \quad & \sum_{r\in R} y_{pr} = 1, && \forall\, p \in P, \\
& \sum_{p\in P(s)} y_{pr} \le C(r), && \forall\, r \in R,\ s \in S_{\text{early}}, \\
& y_{pr} = 0, && \forall\, p \in P,\ r \in R \setminus R(p), \\
& y_{pr_{\text{prev}}(p)} = 1, && \forall\, p \in O, \\
& y_{pr} \le f_{rs} && \forall\, p \in P_F,\ r \in R,\ s \in S_{\text{early}}(p), \\
& y_{pr} \le m_{rs} && \forall\, p \in P_M,\ r \in R,\ s \in S_{\text{early}}(p), \\
& f_{rs} + m_{rs} \le 1 + v^{\text{gm}}_{rs}, && \forall\, r \in R,\ s \in S_{\text{early}}, \\
& y_{pr} \in \{0,1\}, && \forall\, p \in P,\ r \in R, \\
& f_{rs},\, m_{rs} \in \{0,1\}, && \forall\, r \in R,\ s \in S_{\text{early}}, \\
& v^{\text{gm}}_{rs} \in \{0,1\}, && \forall\, r \in R,\ s \in S_{\text{early}}.
\end{aligned}
\tag{A.3}
$$

A.1.4 Full lower bound (partial)

$$
\begin{aligned}
\min \quad & \beta_{\text{cc}} \sum_{n\in N}\sum_{p\in P} a_{np} + \beta_{\text{sv}} \sum_{p\in P}\sum_{s\in S(p)} v^{\text{sv}}_{ps} \\
& + \beta_{\text{wv}} \sum_{n\in N}\sum_{s\in S(n)} v^{\text{wv}}_{ns} + \beta_{\text{wi}} \sum_{s\in S} l^{+}_{s} - l^{-}_{s} \\
\text{s.t.} \quad & \sum_{n\in N(s)} z_{nps} = 1, && \forall\, p \in P,\ s \in S(p), \\
& a_{np} \ge z_{nps}, && \forall\, n \in N,\ p \in P,\ s \in S(n,p), \\
& a_{np} \le \sum_{s\in S(n,p)} z_{nps}, && \forall\, n \in N,\ p \in P, \\
& v^{\text{sv}}_{ps} \ge \sigma_{\text{req}}(p,s) - \sum_{n\in N(s)} \sigma(n)\cdot z_{nps}, && \forall\, p \in P,\ s \in S(p), \\
& \sum_{p\in P(s)} w(p,s)\cdot z_{nps} \le w_{\max}(n,s) + v^{\text{wv}}_{ns}, && \forall\, n \in N,\ s \in S(n), \\
& l^{-}_{s} \le \sum_{p\in P(s)} \frac{w(p,s)}{w_{\max}(n,s)} \cdot z_{nps}, && \forall\, n \in N,\ s \in S(n), \\
& l^{+}_{s} \ge \sum_{p\in P(s)} \frac{w(p,s)}{w_{\max}(n,s)} \cdot z_{nps}, && \forall\, n \in N,\ s \in S(n), \\
& z_{nps} \in \{0,1\}, && \forall\, n \in N,\ p \in P,\ s \in S(n,p), \\
& a_{np} \in \{0,1\}, && \forall\, n \in N,\ p \in P, \\
& v^{\text{sv}}_{ps} \ge 0, && \forall\, p \in P,\ s \in S(p), \\
& v^{\text{wv}}_{ns} \ge 0, && \forall\, n \in N,\ s \in S(n), \\
& l^{-}_{s},\, l^{+}_{s} \ge 0, && \forall\, s \in S,
\end{aligned}
\tag{A.4}
$$

A.2 Upper bounds

A.2.1 Continuity of care (partial)

$$
\begin{aligned}
\max \quad & \sum_{n\in N}\sum_{p\in P} a_{np} \\
\text{s.t.} \quad & \sum_{n\in N(s)} z_{nps} = 1, && \forall\, p \in P,\, s \in S(p), \\
& a_{np} \geq z_{nps}, && \forall\, n \in N,\, p \in P,\, s \in S(n,p), \\
& a_{np} \leq \sum_{s\in S(n,p)} z_{nps}, && \forall\, n \in N,\, p \in P, \\
& z_{nps} \in \{0,1\}, && \forall\, n \in N,\, p \in P,\, s \in S(n,p), \\
& a_{np} \in \{0,1\}, && \forall\, n \in N,\, p \in P.
\end{aligned} \tag{A.5}
$$

A.2.2 Gender-mixing (partial)

$$
\begin{aligned}
\max \quad & \sum_{r\in R}\sum_{s\in S_{\text{early}}} v_{rs}^{\text{gm}} \\
\text{s.t.} \quad & \sum_{r\in R} y_{pr} = 1, && \forall\, p \in P, \\
& \sum_{p\in P(s)} y_{pr} \leq C(r), && \forall\, r \in R,\, s \in S_{\text{early}}, \\
& y_{pr} = 0, && \forall\, p \in P,\, r \in R \setminus R(p), \\
& y_{pr_{\text{prev}}(p)} = 1, && \forall\, p \in O, \\
& y_{pr} \leq f_{rs}, && \forall\, p \in P_F,\, r \in R,\, s \in S_{\text{early}}(p), \\
& y_{pr} \leq m_{rs}, && \forall\, p \in P_M,\, r \in R,\, s \in S_{\text{early}}(p), \\
& f_{rs} \leq \sum_{p\in P_F(s)} y_{pr}, && \forall\, r \in R,\, s \in S_{\text{early}}(p), \\
& m_{rs} \leq \sum_{p\in P_M(s)} y_{pr}, && \forall\, r \in R,\, s \in S_{\text{early}}(p), \\
& f_{rs} + m_{rs} \leq 1 + v_{rs}^{\text{gm}}, && \forall\, r \in R,\, s \in S_{\text{early}}, \\
& v_{rs}^{\text{gm}} \leq f_{rs}, && \forall\, r \in R,\, s \in S_{\text{early}}, \\
& v_{rs}^{\text{gm}} \leq m_{rs}, && \forall\, r \in R,\, s \in S_{\text{early}}, \\
& y_{pr} \in \{0,1\}, && \forall\, p \in P,\, r \in R, \\
& f_{rs},\, m_{rs} \in \{0,1\}, && \forall\, r \in R,\, s \in S_{\text{early}}, \\
& v_{rs}^{\text{gm}} \in \{0,1\}, && \forall\, r \in R,\, s \in S_{\text{early}}.
\end{aligned} \tag{A.6}
$$

A.2.3 Full upper bound (LP)

$$
\begin{aligned}
\max \quad & \beta_{\text{cc}} \sum_{n \in N} \sum_{p \in P} a_{np} + \beta_{\text{gm}} \sum_{r \in R} \sum_{s \in S_{\text{early}}} v^{\text{gm}}_{rs} \\
& + \beta_{\text{sv}} \sum_{p \in P} \sum_{s \in S(p)} v^{\text{sv}}_{ps} + \beta_{\text{wv}} \sum_{n \in N} \sum_{s \in S(n)} v^{\text{wv}}_{ns} \\
& + \beta_{\text{wi}} \sum_{s \in S} l^{+}_{s} - l^{-}_{s} \\
\text{s.t.} \quad & \sum_{r \in R} y_{pr} = 1, && \forall\, p \in P, \\
& \sum_{p \in P(s)} y_{pr} \leq C(r), && \forall\, r \in R,\, s \in S_{\text{early}}, \\
& y_{pr} = 0, && \forall\, p \in P,\, r \in R \setminus R(p), \\
& y_{pr_{\text{prev}}(p)} = 1, && \forall\, p \in O, \\
& \sum_{n \in N(s)} x_{nrs} = 1, && \forall\, r \in R,\, s \in S, \\
& z_{nps} \geq x_{nrs} + y_{pr} - 1, && \forall\, n \in N,\, p \in P, \\
& && \quad r \in R,\, s \in S(n,p), \\
& \sum_{n \in N(s)} z_{nps} = 1, && \forall\, p \in P,\, s \in S(p), \\
& a_{np} \geq z_{nps}, && \forall\, n \in N,\, p \in P, \\
& && \quad s \in S(n,p), \\
& a_{np} \leq \sum_{s \in S(n,p)} z_{nps}, && \forall\, n \in N,\, p \in P, \\
& f_{rs} \leq \sum_{p \in P_f(s)} y_{pr}, && \forall\, r \in R,\, s \in S_{\text{early}}, \\
& m_{rs} \leq \sum_{p \in P_m(s)} y_{pr}, && \forall\, r \in R,\, s \in S_{\text{early}}, \\
& v^{\text{gm}}_{rs} \leq f_{rs}, && \forall\, r \in R,\, s \in S_{\text{early}}, \\
& v^{\text{gm}}_{rs} \leq m_{rs}, && \forall\, r \in R,\, s \in S_{\text{early}}, \\
& v^{\text{sv}}_{ps} = \sum_{n \in N(s)} \max\{0,\, \sigma_{\text{req}}(p,s) - \sigma(n)\} \cdot z_{nps}, && \forall\, p \in P,\, s \in S(p), \\
& v^{\text{wv}}_{ns} \leq M^{\text{wv}}_{ns} \cdot \delta^{\text{wv}}_{ns}, && \forall\, n \in N,\, s \in S(n), \\
& v^{\text{wv}}_{ns} \leq \sum_{p \in P(s)} (w(p,s) \cdot z_{nps}) - w_{\max}(n,s) \\
& \qquad + M^{\text{wv}}_{ns} \cdot (1 - \delta^{\text{wv}}_{ns}), && \forall\, n \in N,\, s \in S(n), \\
& \sum_{n \in N(s)} v^{\text{wv}}_{ns} \leq \max\Big\{0,\, \sum_{p \in P(s)} w(p,s) \\
& \qquad - \min_{n \in N(s)} w_{\max}(n,s)\Big\}, && \forall\, s \in S, \\
& \sum_{n \in N(s)} \delta^{-}_{ns} = 1, && \forall\, s \in S, \\
& \sum_{n \in N(s)} \delta^{+}_{ns} = 1, && \forall\, s \in S,
\end{aligned}
\tag{A.7}
$$

$$
\begin{aligned}
l_s^- \geq & \sum_{p \in P(s)} \frac{w(p,s)}{w_{\max}(n,s)} \cdot z_{nps} && \\
& - M_s^{\mathrm{wi}}(1 - \delta_{ns}^-), && \forall\, s \in S, \\
l_s^+ \leq & \sum_{p \in P(s)} \frac{w(p,s)}{w_{\max}(n,s)} \cdot z_{nps} && \\
& + M_s^{\mathrm{wi}}(1 - \delta_{ns}^+), && \forall\, s \in S, \\
0 \leq\ & x_{nrs} \leq 1, && \forall\, n \in N,\, r \in R,\, s \in S(n), \\
0 \leq\ & y_{pr} \leq 1, && \forall\, p \in P,\, r \in R,\, s \in S_{\mathrm{early}}(p) \\
0 \leq\ & z_{nps} \leq 1, && \forall\, n \in N,\, p \in P,\, s \in S(n,p), \\
0 \leq\ & f_{rs} \leq 1, && \forall\, r \in R,\, s \in S_{\mathrm{early}}, \\
0 \leq\ & m_{rs} \leq 1, && \forall\, r \in R,\, s \in S_{\mathrm{early}}, \\
0 \leq\ & a_{np} \leq 1, && \forall\, n \in N,\, p \in P, \\
0 \leq\ & v_{rs}^{\mathrm{gm}} \leq 1, && \forall\, r \in R,\, s \in S_{\mathrm{early}}, \\
0 \leq\ & \delta_{ns}^{\mathrm{wv}} \leq 1, && \forall\, n \in N,\, s \in S(n), \\
0 \leq\ & \delta_{ns}^- \leq 1, && \forall\, n \in N,\, s \in S(n), \\
0 \leq\ & \delta_{ns}^+ \leq 1, && \forall\, n \in N,\, s \in S(n), \\
& v_{ps}^{\mathrm{sv}} \geq 0, && \forall\, p \in P,\, s \in S(p), \\
& v_{ns}^{\mathrm{wv}} \geq 0, && \forall\, n \in N,\, s \in S(n), \\
& l_s^-,\, l_s^+ \geq 0, && \forall\, s \in S.
\end{aligned}
\qquad \text{(A.7 cont'd)}
$$

A.2.4 Full upper bound (partial)

$$
\begin{aligned}
\max \quad & \beta_{\mathrm{cc}} \sum_{n\in N}\sum_{p\in P} a_{np} + \beta_{\mathrm{sv}} \sum_{p\in P}\sum_{s\in S(p)} v^{\mathrm{sv}}_{ps} \\
& + \beta_{\mathrm{wv}} \textstyle\sum_{n\in N}\sum_{s\in S(n)} v^{\mathrm{wv}}_{ns} + \beta_{\mathrm{wi}} \sum_{s\in S} l^{+}_{s} - l^{-}_{s} \\
\text{s.t.} \quad & \sum_{n\in N(s)} z_{nps} = 1, && \forall\, p\in P,\, s\in S(p), \\
& a_{np} \geq z_{nps}, && \forall\, n\in N,\, p\in P,\, s\in S(n,p), \\
& a_{np} \leq \sum_{s\in S(n,p)} z_{nps}, && \forall\, n\in N,\, p\in P, \\
& v^{\mathrm{sv}}_{ps} = \sum_{n\in N(s)} \max\{0,\ \sigma_{\mathrm{req}}(p,s) - \sigma(n)\}\cdot z_{nps}, && \forall\, p\in P,\, s\in S(p), \\
& v^{\mathrm{wv}}_{ns} \leq M^{\mathrm{wv}}_{ns}\cdot \delta^{\mathrm{wv}}_{ns}, && \forall\, n\in N,\, s\in S(n), \\
& v^{\mathrm{wv}}_{ns} \leq \sum_{p\in P(s)} (w(p,s)\cdot z_{nps}) - w_{\max}(n,s) \\
& \qquad + M^{\mathrm{wv}}_{ns}\cdot(1-\delta^{\mathrm{wv}}_{ns}), && \forall\, n\in N,\, s\in S(n), \\
& \sum_{n\in N(s)} v^{\mathrm{wv}}_{ns} \leq \max\Big\{0,\ \sum_{p\in P(s)} w(p,s) \\
& \qquad - \min_{n\in N(s)} w_{\max}(n,s)\Big\}, && \forall\, s\in S, \\
& \sum_{n\in N(s)} \delta^{-}_{ns} = 1, && \forall\, s\in S, \\
& \sum_{n\in N(s)} \delta^{+}_{ns} = 1, && \forall\, s\in S, \\
& l^{-}_{s} \geq \sum_{p\in P(s)} \frac{w(p,s)}{w_{\max}(n,s)}\cdot z_{nps} \\
& \qquad - M^{\mathrm{wi}}_{s}(1-\delta^{-}_{ns}), && \forall\, s\in S, \\
& l^{+}_{s} \leq \sum_{p\in P(s)} \frac{w(p,s)}{w_{\max}(n,s)}\cdot z_{nps} \\
& \qquad + M^{\mathrm{wi}}_{s}(1-\delta^{+}_{ns}), && \forall\, s\in S, \\
& z_{nps}\in\{0,1\}, && \forall\, n\in N,\, p\in P,\, s\in S(n,p), \\
& a_{np}\in\{0,1\}, && \forall\, n\in N,\, p\in P, \\
& \delta^{\mathrm{wv}}_{ns}\in\{0,1\}, && \forall\, n\in N,\, s\in S(n), \\
& \delta^{-}_{ns},\, \delta^{+}_{ns}\in\{0,1\}, && \forall\, n\in N,\, s\in S(n), \\
& v^{\mathrm{sv}}_{ps}\geq 0, && \forall\, p\in P,\, s\in S(p), \\
& v^{\mathrm{wv}}_{ns}\geq 0, && \forall\, n\in N,\, s\in S(n), \\
& l^{-}_{s},\, l^{+}_{s}\geq 0, && \forall\, s\in S.
\end{aligned}
\tag{A.8}
$$

A.3 ILPs used in heuristics

A.3.1 Non-sequential nurse assignment

$$
\begin{aligned}
\max \quad & \beta_{\mathrm{cc}} \sum_{n\in N}\sum_{p\in P} a_{np} + \beta_{\mathrm{sv}} \sum_{p\in P}\sum_{s\in S(p)} v^{\mathrm{sv}}_{ps} \\
& + \beta_{\mathrm{wv}} \textstyle\sum_{n\in N}\sum_{s\in S(n)} v^{\mathrm{wv}}_{ns} + \beta_{\mathrm{wi}} \sum_{s\in S} l^{+}_{s} - l^{-}_{s} \\
\text{s.t.} \quad & \sum_{n\in N(s)} x_{nrs} = 1, && \forall\, r\in R,\, s\in S, \\
& z_{nps} = x_{nr(p)s}, && \forall\, n\in N,\, p\in P,\, s\in S(n,p), \\
& \sum_{n\in N(s)} z_{nps} = 1, && \forall\, p\in P,\, s\in S(p), \\
& a_{np} \geq z_{nps}, && \forall\, n\in N,\, p\in P,\, s\in S(n,p), \\
& a_{np} \leq \sum_{s\in S(n,p)} z_{nps}, && \forall\, n\in N,\, p\in P, \\
& v^{\mathrm{sv}}_{ps} \geq \sigma_{\mathrm{req}}(p,s) - \sum_{n\in N(s)} \sigma(n)\cdot z_{nps}, && \forall\, p\in P,\, s\in S(p), \\
& \sum_{p\in P(s)} w(p,s)\cdot z_{nps} \leq w_{\max}(n,s) + v^{\mathrm{wv}}_{ns}, && \forall\, n\in N,\, s\in S(n), \\
& l^{-}_{s} \leq \sum_{p\in P(s)} \frac{w(p,s)}{w_{\max}(n,s)}\cdot z_{nps}, && \forall\, n\in N,\, s\in S(n), \\
& l^{+}_{s} \geq \sum_{p\in P(s)} \frac{w(p,s)}{w_{\max}(n,s)}\cdot z_{nps}, && \forall\, n\in N,\, s\in S(n), \\
& x_{nrs} \in \{0,1\}, && \forall\, n\in N,\, r\in R,\, s\in S(n), \\
& z_{nps} \in \{0,1\}, && \forall\, n\in N,\, p\in P, \\
& && \quad s\in S(n,p), \\
& a_{np} \in \{0,1\}, && \forall\, n\in N,\, p\in P, \\
& v^{\mathrm{sv}}_{ps} \geq 0, && \forall\, p\in P,\, s\in S(p), \\
& v^{\mathrm{wv}}_{ns} \geq 0, && \forall\, n\in N,\, s\in S(n), \\
& l^{-}_{s},\, l^{+}_{s} \geq 0, && \forall\, s\in S.
\end{aligned}
\tag{A.9}
$$

A.3.2 Sequential nurse assignment

$$
\begin{aligned}
\max \quad & \beta_{\text{cc}} \sum_{n \in N(s)} \sum_{p \in P(s)} a_{np} + \beta_{\text{sv}} \sum_{p \in P(s)} v_{ps}^{\text{sv}} \\
& + \beta_{\text{wv}} \sum_{n \in N(s)} v_{ns}^{\text{wv}} + \beta_{\text{wi}} \left(l_s^+ - l_s^- \right) \\
\text{s.t.} \quad & \sum_{n \in N(s)} x_{nrs} = 1, && \forall\, r \in R, \\
& z_{nps} = x_{nr(p)s}, && \forall\, n \in N(s),\, p \in P(s), \\
& a_{np} = \begin{cases} 1, & \text{if } n \in \mathcal{N}_s(p), \\ z_{nps}, & \text{otherwise}, \end{cases} && \forall\, n \in N(s),\, p \in P(s), \\
& v_{ps}^{\text{sv}} \geq \sigma_{\text{req}}(p,s) - \sum_{n \in N(s)} \sigma(n) \cdot z_{nps}, && \forall\, p \in P(s), \\
& \sum_{p \in P(s)} w(p,s) \cdot z_{nps} \leq w_{\max}(n,s) + v_{ns}^{\text{wv}}, && \forall\, n \in N(s), \\
& l_s^- \leq \sum_{p \in P(s)} \frac{w(p,s)}{w_{\max}(n,s)} \cdot z_{nps} && \forall\, n \in N(s), \\
& l_s^+ \geq \sum_{p \in P(s)} \frac{w(p,s)}{w_{\max}(n,s)} \cdot z_{nps} && \forall\, n \in N(s), \\
& x_{nrs} \in \{0,1\}, && \forall\, n \in N(s),\, r \in R, \\
& z_{nps} \in \{0,1\}, && \forall\, n \in N(s),\, p \in P(s), \\
& a_{np} \in \{0,1\}, && \forall\, n \in N(s),\, p \in P(s), \\
& v_{ps}^{\text{sv}} \geq 0, && \forall\, p \in P(s), \\
& v_{ns}^{\text{wv}} \geq 0, && \forall\, n \in N(s), \\
& l_s^-,\, l_s^+ \geq 0.
\end{aligned}
\tag{A.10}
$$

A.3.3 Interval heuristic

$$
\begin{aligned}
\max \quad & \beta_{\text{cc}} \sum_{n\in N^k}\sum_{p\in P^k} a_{np} + \beta_{\text{gm}} \sum_{r\in R}\sum_{s\in S^k_{\text{early}}} v^{\text{gm}}_{rs} + \beta_{\text{sv}} \sum_{p\in P^k}\sum_{s\in S^k(p)} v^{\text{sv}}_{ps} \\
& + \beta_{\text{wv}} \sum_{n\in N^k}\sum_{s\in S^k(n)} v^{\text{wv}}_{ns} + \beta_{\text{wi}} \sum_{s\in S^k} l^+_s - l^-_s \\
\text{s.t.} \quad & \sum_{r\in R} y_{pr} = 1, && \forall\, p \in P^k, \\
& \sum_{p\in P(s)} y_{pr} \le C(r), && \forall\, r \in R,\ s \in S^k_{\text{early}}, \\
& y_{pr} = 0, && \forall\, p \in P^k,\ r \in R \setminus R(p), \\
& y_{pr_{k-1}(p)} = 1, && \forall\, p \in P^{k-1} \cap P^k, \\
& \sum_{n\in N(s)} x_{nrs} = 1, && \forall\, r \in R,\ s \in S^k, \\
& z_{nps} \ge x_{nrs} + y_{pr} - 1, && \forall\, n \in N^k,\ p \in P^k, \\
& && \quad r \in R,\ s \in S^k(n,p), \\
& \sum_{n\in N(s)} z_{nps} = 1, && \forall\, s \in S^k,\ p \in P(s), \\
& a_{np} = 1, && \forall\, p \in P^k,\ n \in \mathcal{N}_k(p), \\
& a_{np} \ge z_{nps}, && \forall\, n \in N^k \setminus \mathcal{N}_k(p),\ p \in P^k, \\
& && \quad s \in S^k(n,p), \\
& a_{np} \le \sum_{s\in S^k(n,p)} z_{nps}, && \forall\, n \in N^k \setminus \mathcal{N}_k(p),\ p \in P^k, \\
& y_{pr} \le f_{rs}, && \forall\, p \in P^k_F,\ r \in R,\ s \in S^k_{\text{early}}(p), \\
& y_{pr} \le m_{rs}, && \forall\, p \in P^k_M,\ r \in R,\ s \in S^k_{\text{early}}(p), \\
& f_{rs} + m_{rs} \le 1 + v^{\text{gm}}_{rs}, && \forall\, r \in R,\ s \in s \in S^k_{\text{early}}, \\
& v^{\text{sv}}_{ps} \ge \sigma_{\text{req}}(p,s) - \sum_{n\in N(s)} \sigma(n)\cdot z_{nps}, && \forall\, p \in P^k,\ s \in S^k(p), \\
& \sum_{p\in P^k(s)} w(p,s)\cdot z_{nps} \le w_{\max}(n,s) + v^{\text{wv}}_{ns}, && \forall\, n \in N^k,\ s \in S^k(n), \\
& l^-_s \le \sum_{p\in P^k(s)} \frac{w(p,s)}{w_{\max}(p,s)}\cdot z_{nps}, && \forall\, n \in N^k,\ s \in S^k(n), \\
& l^+_s \ge \sum_{p\in P^k(s)} \frac{w(p,s)}{w_{\max}(p,s)}\cdot z_{nps}, && \forall\, n \in N^k,\ s \in S^k(n), \\
& x_{nrs} \in \{0,1\}, && \forall\, n \in N^k,\ r \in R,\ s \in S^k(n), \\
& y_{pr} \in \{0,1\}, && \forall\, p \in P^k,\ r \in R, \\
& z_{nps} \in \{0,1\}, && \forall\, n \in N^k,\ p \in P^k,\ s \in S^k(n,p), \\
& a_{np} \in \{0,1\}, && \forall\, n \in N^k,\ p \in P^k \\
& f_{rs},\ m_{rs} \in \{0,1\}, && \forall\, r \in R,\ s \in S^k_{\text{early}}, \\
& v^{\text{gm}}_{rs} \in \{0,1\}, && \forall\, r \in R,\ s \in S^k_{\text{early}}, \\
& v^{\text{sv}}_{ps} \ge 0, && \forall\, p \in P^k,\ s \in S^k(p), \\
& v^{\text{wv}}_{ns} \ge 0, && \forall\, n \in N^k,\ s \in S^k(n), \\
& l^-_s,\ l^+_s \ge 0, && \forall\, s \in S^k.
\end{aligned}
\tag{A.11}
$$

A.4 ILPs used in online heuristics

A.4.1 Non-sequential nurse assignment

$$
\begin{aligned}
\max \quad & \beta_{\text{cc}} \sum_{n\in N}\sum_{p\in P} a_{np} + \beta_{\text{sv}} \sum_{p\in P}\sum_{s\in S(p)} v^{\text{sv}}_{ps} \\
& + \beta_{\text{wv}} \textstyle\sum_{n\in N}\sum_{s\in S(n)} v^{\text{wv}}_{ns} + \beta_{\text{wi}} \sum_{s\in S} l^{+}_{s} - l^{-}_{s} \\
\text{s.t.} \quad & \sum_{n\in N(s)} x_{nrs} = 1, && \forall\, r\in R,\, s\in S, \\
& z_{nps} = x_{nr(p)s}, && \forall\, n\in N,\, p\in P,\, s\in S(n,p), \\
& \sum_{n\in N(s)} z_{nps} = 1, && \forall\, p\in P,\, s\in S(p), \\
& a_{np} = 1, && \forall\, p\in O,\, n\in \mathcal{N}_{\text{prev}}(p), \\
& a_{np} \geq z_{nps}, && \forall\, n\in N,\, p\in P,\, s\in S(n,p), \\
& a_{np} \leq \sum_{s\in S(n,p)} z_{nps}, && \forall\, n\in N,\, p\in P, \\
& v^{\text{sv}}_{ps} \geq \sigma_{\text{req}}(p,s) - \sum_{n\in N(s)} \sigma(n)\cdot z_{nps}, && \forall\, p\in P,\, s\in S(p), \\
& \sum_{p\in P(s)} w(p,s)\cdot z_{nps} \leq w_{\max}(n,s) + v^{\text{wv}}_{ns}, && \forall\, n\in N,\, s\in S(n), \\
& l^{-}_{s} \leq \sum_{p\in P(s)} \frac{w(p,s)}{w_{\max}(n,s)}\cdot z_{nps}, && \forall\, n\in N,\, s\in S(n), \\
& l^{+}_{s} \geq \sum_{p\in P(s)} \frac{w(p,s)}{w_{\max}(n,s)}\cdot z_{nps}, && \forall\, n\in N,\, s\in S(n), \\
& x_{nrs} \in \{0,1\}, && \forall\, n\in N,\, r\in R,\, s\in S(n), \\
& z_{nps} \in \{0,1\}, && \forall\, n\in N,\, p\in P,\, s\in S(n,p), \\
& a_{np} \in \{0,1\}, && \forall\, n\in N,\, p\in P, \\
& v^{\text{sv}}_{ps} \geq 0, && \forall\, p\in P,\, s\in S(p), \\
& v^{\text{wv}}_{ns} \geq 0, && \forall\, n\in N,\, s\in S(n), \\
& l^{-}_{s},\, l^{+}_{s} \geq 0, && \forall\, s\in S.
\end{aligned}
\tag{A.12}
$$

A.4.2 Interval heuristic

$$
\begin{aligned}
\max \quad & \beta_{\mathrm{cc}} \sum_{n\in N^k}\sum_{p\in P^k} a_{np} + \beta_{\mathrm{gm}} \sum_{r\in R}\sum_{s\in S^k_{\mathrm{early}}} v^{\mathrm{gm}}_{rs} + \beta_{\mathrm{sv}} \sum_{p\in P^k}\sum_{s\in S^k(p)} v^{\mathrm{sv}}_{ps} \\
& + \beta_{\mathrm{wv}} \sum_{n\in N^k}\sum_{s\in S^k(n)} v^{\mathrm{wv}}_{ns} + \beta_{\mathrm{wi}} \sum_{s\in S^k} l^+_s - l^-_s + \beta_{\mathrm{tr}} \sum_{\substack{p\in O \\ \text{if } k=1}} v^{\mathrm{tr}}_p \\
\text{s.t.} \quad & \sum_{r\in R} y_{pr} = 1, && \forall\, p\in P^k, \\
& \sum_{p\in P(s)} y_{pr} \le C(r), && \forall\, r\in R,\ s\in S^k_{\mathrm{early}}, \\
& y_{pr} = 0, && \forall\, p\in P^k,\ r\in R\setminus R(p), \\
& y_{pr_{k-1}(p)} = \begin{cases} 1-v^{\mathrm{tr}}_p, & \text{if } k=1 \\ 1, & \text{otherwise,} \end{cases} && \forall\, p\in P^{k-1}\cap P^k, \\
& \sum_{n\in N(s)} x_{nrs} = 1, && \forall\, r\in R,\ s\in S^k, \\
& z_{nps} \ge x_{nrs} + y_{pr} - 1, && \forall\, n\in N^k,\ p\in P^k, \\
& && \quad r\in R,\ s\in S^k(n,p), \\
& \sum_{n\in N(s)} z_{nps} = 1, && \forall\, s\in S^k,\ p\in P(s), \\
& a_{np} = 1, && \forall\, p\in P^k,\ n\in \mathcal{N}_k(p), \\
& a_{np} \ge z_{nps}, && \forall\, n\in N^k\setminus\mathcal{N}_k(p),\ p\in P^k, \\
& && \quad s\in S^k(n,p), \\
& a_{np} \le \sum_{s\in S^k(n,p)} z_{nps}, && \forall\, n\in N^k\setminus\mathcal{N}_k(p),\ p\in P^k, \\
& y_{pr} \le f_{rs}, && \forall\, p\in P^k_F,\ r\in R,\ s\in S^k_{\mathrm{early}}(p), \\
& y_{pr} \le m_{rs}, && \forall\, p\in P^k_M,\ r\in R,\ s\in S^k_{\mathrm{early}}(p), \\
& f_{rs} + m_{rs} \le 1 + v^{\mathrm{gm}}_{rs}, && \forall\, r\in R,\ s\in s\in S^k_{\mathrm{early}}, \\
& v^{\mathrm{sv}}_{ps} \ge \sigma_{\mathrm{req}}(p,s) - \sum_{n\in N(s)} \sigma(n)\cdot z_{nps}, && \forall\, p\in P^k,\ s\in S^k(p), \\
& \sum_{p\in P^k(s)} w(p,s)\cdot z_{nps} \le w_{\max}(n,s) + v^{\mathrm{wv}}_{ns}, && \forall\, n\in N^k,\ s\in S^k(n), \\
& l^-_s \le \sum_{p\in P^k(s)} \frac{w(p,s)}{w_{\max}(p,s)}\cdot z_{nps}, && \forall\, n\in N^k,\ s\in S^k(n), \\
& l^+_s \ge \sum_{p\in P^k(s)} \frac{w(p,s)}{w_{\max}(p,s)}\cdot z_{nps}, && \forall\, n\in N^k,\ s\in S^k(n), \\
& x_{nrs} \in \{0,1\}, && \forall\, n\in N^k,\ r\in R,\ s\in S^k(n), \\
& y_{pr} \in \{0,1\}, && \forall\, p\in P^k,\ r\in R, \\
& z_{nps} \in \{0,1\}, && \forall\, n\in N^k,\ p\in P^k,\ s\in S^k(n,p), \\
& a_{np} \in \{0,1\}, && \forall\, n\in N^k,\ p\in P^k \\
& f_{rs},\, m_{rs} \in \{0,1\}, && \forall\, r\in R,\ s\in S^k_{\mathrm{early}}, \\
& v^{\mathrm{gm}}_{rs} \in \{0,1\}, && \forall\, r\in R,\ s\in S^k_{\mathrm{early}}, \\
& v^{\mathrm{tr}}_p\ \{0,1\}, && \forall\, p\in O, \text{ if } k=1, \\
& v^{\mathrm{sv}}_{ps} \ge 0, && \forall\, p\in P^k,\ s\in S^k(p), \\
& v^{\mathrm{wv}}_{ns} \ge 0, && \forall\, n\in N^k,\ s\in S^k(n), \\
& l^-_s,\, l^+_s \ge 0, && \forall\, s\in S^k.
\end{aligned}
\tag{A.13}
$$

Appendix B

Additional Tables

B.1 Lower and upper bounds

Table B.1: Lower and upper bounds for the public instances. Obtained using the methods described in Section 4.2.

Instance	Lower bound	Upper bound	Instance	Lower bound	Upper bound
i01	146.167	537.300	m01	718.233	2187.483
i02	240.533	1056.183	m02	253.017	1071.067
i03	156.492	510.983	m03	229.565	895.167
i04	446.717	2605.717	m04	318.050	1824.183
i05	339.933	2457.800	m05	366.017	2320.825
i06	279.100	1504.117	m06	620.500	5051.581
i07	461.950	3040.383	m07	443.683	2457.083
i08	1056.533	8782.967	m08	741.250	6953.084
i09	542.983	3957.533	m09	1342.867	10592.111
i10	717.500	3796.967	m10	798.900	5297.927
i11	383.767	2605.700	m11	910.733	10110.279
i12	652.067	4039.050	m12	1080.567	8183.398
i13	794.083	4368.667	m13	971.600	9181.217
i14	652.100	5206.200	m14	816.267	8072.276
i15	684.750	5535.917	m15	1956.217	14280.154
i16	902.317	5973.267	m16	953.450	9072.383
i17	1509.717	12027.083	m17	980.800	7745.107
i18	731.017	6329.000	m18	1103.783	11163.876
i19	1201.950	12168.350	m19	970.383	7268.533
i20	811.700	5888.933	m20	1029.400	11828.857
i21	1332.450	10669.700	m21	1213.517	9057.271
i22	2038.133	14815.600	m22	976.567	8718.800
i23	1456.650	14750.100	m23	1194.817	11508.000
i24	1647.117	16938.500	m24	1709.117	21845.150
i25	1032.351	9897.633	m25	1354.367	13924.623
i26	2040.700	17591.100	m26	2098.800	27158.386
i27	2276.433	18404.900	m27	1168.600	12213.364
i28	1085.300	10941.500	m28	2246.933	24579.000
i29	1380.717	11326.300	m29	1844.733	25338.072
i30	1703.818	14398.500	m30	2197.218	29465.024

B.2 Patient-to-room room ordering

Table B.2: Objective values for different room orderings in the greedy patient-to-room assignment heuristic. The * next to the instance name indicates that the nurse-to-room assignment ILP might not be optimal, i.e., it was terminated after five minutes.

Instance	OG	GO	GS_0	$GS_{0.5}$	$GS_{0.8}$	GS_1	GOS_0
test01	269.667	254.650	259.267	259.267	258.600	257.217	**253.317**
test02*	383.833	376.400	377.850	**371.283**	373.350	372.583	382.783
test03	147.233	146.317	153.433	152.967	148.683	**142.633**	142.750
test04*	413.050	401.183	415.050	406.533	386.267	**382.633**	403.700
test05	191.767	184.083	201.667	201.500	193.567	**180.633**	184.283
i01	168.800	165.667	172.467	172.467	172.467	**164.650**	165.667
i02	304.783	287.100	293.900	291.583	291.900	288.700	**286.383**
i03	192.217	**177.617**	177.900	177.900	178.883	181.783	178.650
i04*	621.750	597.217	610.550	603.783	585.533	**584.017**	596.767
i05*	381.450	372.183	411.617	403.767	392.933	370.217	**369.400**
i06	332.050	320.617	334.183	333.317	328.300	**320.367**	320.617
test06*	514.550	484.967	463.917	**461.117**	472.267	473.683	463.650
test07*	915.600	858.933	891.600	883.217	863.233	**856.150**	863.117
test08*	686.583	623.450	622.550	619.917	**619.867**	620.900	620.933
test09*	891.967	837.650	876.750	852.767	802.033	**794.450**	830.300
i07*	606.550	556.133	582.250	572.267	548.617	**538.917**	559.517
i08*	1792.083	1627.117	1645.983	**1606.950**	1607.583	1632.783	1627.083
i09*	825.533	789.033	**784.333**	796.200	784.450	792.067	788.833
i10*	983.400	924.817	914.733	915.683	**903.450**	909.467	924.650
i11*	446.883	428.883	479.250	469.967	442.267	**418.350**	429.000
i12*	951.667	895.183	920.217	889.333	887.983	**878.200**	900.117
i13*	1187.900	**1126.867**	1157.517	1165.917	1144.067	1153.783	1155.617
i14*	889.550	858.883	888.617	873.567	828.733	**826.600**	863.317
i15*	1052.283	1018.200	994.200	996.833	**941.850**	965.217	1013.000
i16*	1400.183	1355.633	**1311.867**	1328.483	1328.217	1325.183	1356.617
i18*	1076.383	1008.533	1028.983	995.467	**963.583**	965.283	1001.333
i20*	1311.617	1249.567	1229.083	1235.183	**1210.117**	1212.850	1267.400
test10*	3491.900	3359.167	3432.300	3253.567	**3103.700**	3205.467	3367.017
i17*	2474.067	2295.167	2326.183	2249.667	**2198.317**	2212.517	2306.283
i19*	2013.467	1882.300	1895.933	1817.767	**1784.967**	1806.900	1897.633
i21*	2147.100	2032.217	1978.367	1928.033	**1925.867**	1937.317	2021.567
i22*	3395.183	3216.983	3142.350	**3107.900**	3157.250	3127.317	3197.850
i23*	2456.183	2324.667	2310.733	2241.017	**2117.550**	2180.233	2296.017
i24*	2619.367	2499.500	2641.133	2529.000	**2350.433**	2381.100	2481.817
i25*	1688.583	1588.317	1596.583	1557.933	1527.133	**1514.300**	1596.300
i26*	3469.033	3172.550	3186.383	3111.883	3075.167	**3069.983**	3199.250
i27*	4023.967	3776.983	3729.767	3637.433	**3548.467**	3612.267	3721.483
i28*	1732.700	1634.650	1692.517	1592.683	1524.717	**1506.533**	1603.817
i29*	2444.333	2281.833	2217.450	**2186.617**	2195.383	2224.033	2258.417
i30*	2970.483	2861.533	2810.700	2751.500	**2712.417**	2739.633	2874.700

B.3 Nurse-to-room shift ordering

Table B.3: Objective values for different shift orderings in the nurse-to-room assignment heuristic. Reports the average over 10 runs.

Instance	Chronological		Nurses		Patients		Total workload		Average workload	
	ascend.	descend.	ascend.	descend.	ascend.	descend.	ascend.	descend.	ascend.	descend.
test01	286.678	**279.482**	282.663	290.550	293.823	279.722	296.413	286.235	296.717	290.230
test02	**405.848**	408.977	411.877	417.538	417.732	411.478	421.062	416.590	415.798	411.468
test03	166.423	163.383	161.785	167.300	161.707	163.807	166.092	162.345	167.260	**160.115**
test04	438.507	432.417	440.372	442.630	447.975	**430.508**	446.203	437.687	439.555	445.650
test05	208.155	212.163	**204.352**	208.210	217.497	212.823	206.723	210.343	210.878	204.678
i01	181.065	178.480	178.703	177.653	183.873	**173.283**	186.668	173.987	182.223	175.713
i02	**306.070**	316.062	310.802	307.565	321.217	314.762	321.498	316.433	308.607	315.097
i03	192.092	192.807	193.527	193.713	196.440	195.350	198.085	191.575	196.395	**191.178**
i04	643.448	**641.918**	648.987	654.230	654.370	643.863	654.830	645.327	644.910	650.030
i05	456.327	455.195	457.207	469.418	465.078	**453.455**	464.372	462.467	470.983	461.277
i06	373.597	367.555	**359.470**	378.990	379.100	365.617	379.843	361.998	376.225	375.440
test06	538.438	**531.725**	538.418	539.110	544.767	538.412	545.277	538.688	546.888	536.865
test07	945.228	937.858	**930.002**	945.423	951.952	942.980	942.562	932.575	942.592	948.708
test08	707.893	708.287	706.428	721.732	714.010	709.910	720.485	**700.587**	713.740	712.508
test09	959.840	963.317	**953.828**	979.840	986.067	954.615	986.632	955.053	967.542	968.752
i07	**647.748**	653.770	655.837	655.380	660.873	650.512	672.488	650.145	656.063	656.207
i08	**1776.593**	1791.028	1789.245	1800.827	1808.507	1788.100	1807.687	1777.880	1802.425	1800.598
i09	862.195	871.792	**861.360**	878.297	891.412	862.360	884.297	868.162	877.447	879.690
i10	**982.730**	988.105	996.305	999.088	1001.805	998.360	1001.307	996.127	1009.138	982.762
i11	548.897	553.720	552.347	550.988	550.733	552.143	558.383	**543.780**	556.178	548.327
i12	972.615	973.943	975.015	980.972	978.990	975.715	976.077	**971.320**	979.402	982.820
i13	**1209.347**	1224.137	1215.210	1224.268	1213.040	1223.788	1223.497	1235.085	1222.937	1225.368
i14	998.532	1010.510	1008.950	1018.743	1019.697	1008.205	1028.267	**994.433**	1016.863	1007.067
i15	1098.790	1089.432	1091.710	1101.487	1117.380	**1087.520**	1117.973	1087.605	1098.828	1092.167
i16	1439.615	1435.588	1437.430	1448.225	1457.722	**1425.735**	1451.448	1433.310	1463.967	1440.913
i18	1181.795	1191.318	1208.232	1184.530	1205.533	1190.990	1217.042	1176.145	1215.753	**1170.017**
i20	1314.982	1313.450	1336.172	1325.583	1339.045	1318.462	1340.933	**1312.883**	1336.285	1325.035
test10	**3559.508**	3598.153	3615.308	3574.965	3617.322	3582.338	3630.238	3570.343	3627.157	3564.450
i17	**2453.682**	2500.618	2492.672	2472.427	2513.175	2484.263	2510.928	2475.537	2502.618	2475.545
i19	2110.845	2105.950	2122.558	2122.708	2148.058	2114.488	2146.493	2103.678	2150.278	**2093.403**
i21	2180.250	2179.850	2212.288	2202.683	2221.218	2186.545	2227.127	**2176.583**	2222.743	2195.668
i22	**3344.743**	3360.623	3377.412	3381.068	3421.028	3381.460	3413.812	3373.607	3402.473	3371.683
i23	**2482.267**	2518.792	2533.658	2511.478	2545.637	2515.290	2551.650	2484.033	2548.247	2499.465
i24	2787.135	2807.867	2837.063	2789.730	2832.990	2803.710	2861.048	**2778.850**	2854.538	2784.447
i25	1762.350	1770.557	1781.308	1762.113	1796.988	1759.540	1789.822	**1750.042**	1785.548	1753.083
i26	**3349.793**	3380.392	3412.600	3383.255	3418.898	3396.597	3428.287	3373.745	3423.507	3385.360
i27	**3850.838**	3886.867	3890.017	3905.100	3927.178	3908.223	3925.697	3885.107	3913.648	3897.472
i28	1827.017	1830.010	1838.318	1832.043	1850.488	1832.093	1866.862	**1816.457**	1874.273	1819.047
i29	**2352.348**	2380.150	2384.262	2381.462	2413.998	2378.955	2421.668	2376.640	2394.718	2382.952
i30	**2932.090**	2934.068	2954.103	2969.715	2994.755	2940.195	3000.397	2946.723	2980.005	2952.828

B.4 Evaluation results

Table B.4: Evaluation results summary (taken from Tables 6.1, 6.2 and 6.3).

Instance	Greedy (seq.)	Greedy (non-seq.)	Interval heuristic	Simulated annealing
m01	906.882	872.033	**824.908**	835.427
m02	323.045	279.950	**274.118**	277.552
m03	289.900	277.183	**258.433**	260.672
m04	471.347	434.700	429.198	**425.912**
m05	499.373	455.283	450.187	**437.525**
m06	924.947	**781.167**	841.562	792.748
m07	643.730	**601.883**	622.025	617.645
m08	1183.157	**982.783**	1069.415	992.088
m09	2177.377	2012.333	2019.463	**1919.398**
m10	1154.470	1045.600	1033.155	**1016.657**
m11	1500.852	**1154.883**	1334.780	1231.040
m12	1686.765	1493.833	1591.500	**1484.527**
m13	1580.527	**1301.000**	1649.868	1353.840
m14	1338.422	**1147.750**	1270.928	1161.270
m16	1663.323	1506.150	1664.422	**1471.692**
m17	1512.165	1349.700	1427.437	**1317.112**
m19	-	-	1603.237	**1518.827**
m21	1986.157	**1847.833**	2095.982	1898.120
m15	3198.123	**2989.917**	3065.888	3057.700
m18	1917.645	**1706.767**	1880.235	1733.962
m20	1805.113	**1503.717**	1781.325	1557.273
m22	1568.878	**1297.350**	1565.772	1331.782
m23	2005.803	1772.217	2097.650	**1763.392**
m24	3154.407	**2809.267**	3145.672	2901.070
m25	2346.198	**2130.933**	2306.762	2172.640
m26	3827.528	**3329.767**	3841.007	3440.133
m27	2003.418	**1753.700**	1978.795	1772.430
m28	3838.658	**3224.450**	4406.160	3412.828
m29	3343.910	**2608.267**	3941.432	2862.192
m30	4203.528	**3916.433**	6072.215	4080.588