universität wien

DISSERTATION / DOCTORAL THESIS

Titel der Dissertation / Title of the Doctoral Thesis

"Solution Techniques for Rich and Real-world Appointment and Staff Scheduling Problems in Health Care"

verfasst von / submitted by

Mag. Petra Vogl, B.Stat. MStat

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Doctor of Philosophy (PhD)

Wien, 2019 / Vienna, 2019

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet	A 794 370 403
Dissertationsgebiet lt. Studienblatt / field of study as it appears on the student record sheet	Wirtschaftswissenschaften (Logistics and Operations Management)
Betreut von / Supervisor:	Univ.-Prof. Mag. Dr. Karl Franz Dörner, Privatdoz.

Acknowledgments

First, I would like to thank my supervisor and advisor, Prof. Karl F. Dörner for giving me the freedom to choose my research topic and guidance when I needed it. I deeply acknowledge that Prof. Dörner encouraged me to return to the University after I have paused my academic career for some years. He not only supported me throughout the past four years with his scientific knowledge but the confidence he had in my research was highly motivating and inspiring.

Additionally, I would like to express my profound gratitude to Dr. Roland Braune, who played an important role in completing this thesis. Roland and I spent hours and hours brainstorming research directions and he would always and anytime lend an ear if I was stuck somewhere on the road to this thesis.

Walter Gutjahr guided and supervised me in addressing uncertainty in schedule optimization. I would like to thank him for his patience, the valuable leads and his open door even after he officially retired from University.

Moreover, I thank EBG MedAustron GmbH (Franz Coreth, Michael Pata and their whole team) as well as the Austrian Red Cross Blood Donation Services (Lars Eberhart and the whole team) for giving me insight into their planning problems and for the fruitful collaboration.

I would like to thank the anonymous reviewers of my papers and the countless feedback I received from the audience when I presented my work at international conferences and when visiting other Universities. Special thanks go to Prof. Günther Raidl, Johannes Maschler, and Martin Riedler, who accepted to work with us on the topic of radiotherapy appointment scheduling.

Special thanks go to my parents, Christine and Josef, for believing in me. Without their constant support, I would not have been able to finish (or even begin) this thesis. I owe special thanks to Sebastian Lehner. He contributed to this thesis more than he knows: By building me up in challenging times, by bringing new ideas to light, and by celebrating achievements with me. Finally, I would like to thank my friends – the ones who are near and the ones who are far – for their support and friendship.

Affidavit

English

I hereby declare that I have authored this thesis independently, that I have not used other than the declared sources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

German

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Vienna/Wien, 24.01.2019 _____

(Signature/Unterschrift)

Abstract (German)

Termin- und Personaleinsatzplanungsprobleme sind allgegenwärtig im Gesundheitswesen: Einerseits müssen Patiententermine innerhalb vordefinierter Zeitfenster so geplant werden, dass etwaige knappe Ressourcen möglichst effizient genutzt werden können und gleichzeitig Patientenwartezeiten gering gehalten werden. Andererseits sind im Gesundheitswesen eine Vielzahl an Mitarbeitern und Mitarbeiterinnen mit unterschiedlicher Ausbildung und unterschiedlichen Vertragsmodellen tätig. Die vorliegende Dissertation behandelt beide genannten Problemkategorien anhand von Echtweltfragestellungen: (1) Die Patiententerminplanung von wiederkehrenden Behandlungen in der (Ionen-)Radiotherapie sowie (2) die Personaleinsatzplanung von Ärzten, Krankenschwestern und medizinischem Hilfspersonal im Blutspendewesen, wo darüber hinaus auch eine Entscheidung getroffen werden muss, welche Aktionen an welchem Tag stattfinden sollen.

Zu Beginn werden die genannten Probleme mathematisch modelliert und mittels kommerziellen Optimierungstools exakt gelöst. Dabei stößt man allerdings bei echtwelt-inspirierten Instanzgrößen rasch an die Grenze der Lösbarkeit. Daher werden maßgeschneiderte Algorithmen entwickelt, welche auch große Instanzen innerhalb geringer Rechenzeit in guter Qualität lösen können. Dabei werden unterschiedliche metaheuristische Verfahren – populationsbasierte Ansätze ebenso wie lokale Suchverfahren sowie die Kombination beider Suchalgorithmen – verglichen und intensiv getestet. Neben deterministischen Ansätzen wird unter anderem gezeigt, dass bei unsicheren Therapiedauern stochastische Methoden eine deutliche Verringerung der Patientenwartezeit bei gleichzeitiger Minimierung der Stillstandzeit teurer Ressourcen bewirken können. Dabei werden unterschiedliche Ansätze zur Abschätzung von Wartezeiten vorgestellt und verglichen.

Abstract (English)

In health care, scheduling and rostering problems are omnipresent: On the one hand, patient appointments need to be scheduled within predefined time windows, such that scarce resources are used as efficiently as possible while simultaneously minimizing patient waiting times. On the other hand, health care facilities need to schedule their employees, who typically feature different skills and contract types. In this PhD thesis both mentioned optimization problem categories are addressed by means of real-world problems: First, we schedule recurring appointments for patients receiving radiotherapy for a newly built ion beam therapy center. Second, for blood collection services that act in the field, skilled staff members need to be scheduled to serve the chosen activities, where typically one needs to decide on which activities should take place on which days additionally.

At first, we model all problems mathematically and use commercial optimization tools to assess their complexity. This approach quickly reaches its limits with real-world problem sizes. Hence, we develop tailor-made optimization algorithms, which are capable of solving even large problem sizes in reasonable quality within short running times. We compare and intensively test different metaheuristical approaches – population-based strategies as well as local search techniques and combinations of those approaches – to tackle the problems. In addition to addressing the problems deterministically, the thesis points out that in case of uncertain appointment durations stochastic methods yield a considerate decrease in patient waiting time when simultaneously minimizing idle time of scarce resources. Here, we propose different (stochastic) techniques to evaluate the quality of a baseline schedule in matters of patient waiting time throughout the optimization.

Preface

This dissertation contains results of research I have conducted over the last four years at the University of Vienna. Parts of the book have already been published in scientific journals and were presented in scientific conferences, while some content is still under revision or in preparation for publication.

More precisely, Chapters 3 to 4 are published in Vogl et al. [2018a,b], while Chapter 5 has been submitted to the renowned "Flexible Services and Manufacturing Journal" (FSMJ) and is currently under review (Vogl et al. [2018c]). In general, results from Part I have been presented in seven international scientific conferences (EULOG 2018, EURO 2018, MIC 2017, MAPSP 2017, EUROCAST 2017, EURO 2016, and GOR HCM 2016).

An extended abstract containing the problem description on Part II is part of the proceedings for the PATAT conference 2018 (Vogl and Braune [2018]). A paper equivalent to Part II is currently in preparation for submission to "Annals or Operations Research", special issue for PATAT 2018 (planned submission end of February 2019, Vogl and Braune [2019]).

Contents

Chapter 1

Introduction

1.1 Scheduling in Health Care

Depleting resources and skilled labor shortage adds pressure to health care institutions, which have been facing rapidly rising demand over the past centuries (Walters [2019]). Cutting costs in health care, however, must not lead to a deterioration of patient care (Lopez [2017]), for example in case of increased waiting time due to tight appointment scheduling. According to Hall [2012], "waiting is the consequence of a mismatch between the needs for service and the availability of resources to provide the service". He identifies two possible causes for this mismatch. Either there are less resources available than are needed. Or a lack of synchronization might cause health care systems to perform inefficiently. Matching resources such as doctors, rooms and machines to needs of patients in health care is complicated by various constraints such as sequences of treatments for patients and specialized work spaces. Additionally, durations of procedures in health care are highly variable, causing waiting times. Hall [2012] admonishes, that forcing patients to wait until a treatment has highly undesired consequences: The patient might be in pain or the patient's condition might worsen during the waiting time.

The main goal in scheduling is to match patient needs and resources. In general health care scheduling problems can be divided into two main classes: (1) Staff scheduling problems and (2) patient appointment scheduling problems: The classical problem within the first category is nurse scheduling, which has already been intensively studied in the literature (De Causmaecker and Van Den Berghe [2011]). Concerning the patient appointment scheduling problem, many different practical applications are known such as appointment scheduling in primary or specialty care clinics (e.g., Cayirli and Veral [2003]), operating room scheduling (e.g., Cardoen et al. [2010]) or radiotherapy scheduling (see Section 4.2 for related work on this topic). The home health care scheduling problem combines the two

subproblems and simultaneously schedules nurses and appointments.

Patient-related scheduling problems are challenging since additional constraints need to be considered. The following list should give a short impression on the special characteristics of health care scheduling problems (see, e.g., Gupta and Denton [2008]):

1. While some health care departments mainly deal with so-called elective (non-emergency) patients and can, therefore, plan the patient appointments and the corresponding capacities needed in advance, other disciplines face a high number of emergency patients who have to be treated promptly.
2. However, even for planned appointments, the treatment duration may vary considerably. For example, the patient could be in a more severe condition than expected and which makes the planned treatment impossible, or the ascertained diagnosis is vague and the concrete treatment might be chosen on the fly.
3. Patient convenience is fundamental. Therefore, for most problems, the patient waiting time should be minimized. This also applies to waiting time in between multiple appointments on a given day. Further, in some cases like emergency patients or critically ill patients, the waiting time to treatment may highly correlate with the probability of recovery.
4. However, not only the time until the appointment is crucial. Especially when treating chronically ill patients, the relationship between the doctor/nurse and the patient is of utmost importance. Therefore so-called "continuity in care" should be achieved. This also applies to the time of day the recurring treatments take place, which should also remain stable.
5. Typically, only a fixed number of resources such as computer tomographies (CTs) or operating rooms are available. Therefore, multiple patients compete for these resources. We deal with this special problem in our first project, where the particle beam of the radiotherapy facility is the bottleneck resource.

1.2 Research Questions

The main focus of the proposed PhD-thesis lies in achieving close-to-optimal solutions for NP-hard scheduling problems with applications in health care management. The proposed projects cover both categories of scheduling problems in health care, namely staff and patient appointment scheduling.

While in classical production scheduling problems the objective function is mainly cost- or efficiency-related, different goals might be followed when scheduling patient appointments or making staffing decisions. These objectives are either present directly within the objective function, or they form new constraints that assure a minimum/maximum level of patient/staff satisfaction. Furthermore, some restrictions are either medically demanded (e.g., minimum of 4 treatments per week) or required by law (e.g., maximum working hours per nurse). The planned projects account for those special features of scheduling problems in health care. In addition, the considered scheduling problems contain additional degrees of freedom which make scheduling even more complex, i.e. for example optional patient appointments in radiotherapy scheduling or flexible activity dates for the blood donation activity staffing problem.

To understand the problems in more detail, a mathematical problem formulation is essential. Therefore this step forms the first task of each subproject. However, the majority of large-scale, real-world scheduling problems including high degrees of freedom are hardly solvable using exact methods. Therefore, the aim of this dissertation is to devise competitive heuristic algorithms to solve the proposed rich scheduling problems. For example, (hybrid) genetic algorithms or local search based heuristics deliver promising results for large-scale problems, as shown in the results sections of the subprojects.

The following list summarizes the research questions:

1. Which objectives and constraints distinguish scheduling problems in health care from classical scheduling problems?
2. Which methods/algorithms deliver good to even optimal results for the proposed rich and real-world scheduling problems at a reasonable computation time?

1.3 Structure of the Thesis

This thesis is organized as follows: Chapter 2 gives a short introduction of the optimization algorithms applied in this thesis and points to various specialized books and book chapters.

Part I deals with the radiotherapy patient appointment scheduling problem, which is already published (Vogl et al. [2018a,b]) or currently under revision (Vogl et al. [2018c]). More precisely, Chapters 3 to 4 introduce the problem of scheduling recurring appointments to patient receiving radiotherapy and solves the deterministic variant of the planning problem (Vogl et al. [2018a,b]). Chapter 5 addresses a slightly modified version stochastically, focusing on the minimization of patient waiting times in case of disruptions (Vogl et al. [2018c]). Chapter 6 concludes the first part and states possible directions for future research on radiotherapy appointment scheduling (as in Vogl et al. [2018a,b,c]).

Part II is dedicated to a staff scheduling problem that arises for example at the Austrian Red Cross blood donation services. Chapter 7 introduces the problem formally and gives a general literature review on combined staff scheduling and activity selection, part of which is available in Vogl and Braune [2018]. Then, Chapter 8 proposes a solution method and states results of extensive computational tests on real-world inspired problem instances. Chapter 9 concludes the second part. A paper equivalent to Part II is currently in preparation for submission to "Annals or Operations Research", special issue for PATAT 2018 (planned submission end of February 2019, the paper in preparation will be cited by Vogl and Braune [2019]).

Chapter 2

Optimization Algorithms

To address complex optimization problems and to find good quality solutions, we follow two main strategies throughout this dissertation. At first, we model and formulate the problem mathematically as a mixed integer linear program (MIP), which can then be solved using commercial tools like CPLEX or Gurobi. We give a short introduction to the corresponding modeling technique in Section 2.1. Then, Section 2.2 concentrates on heuristic search concepts as an alternative to exact optimization approaches with the goal to find good quality solutions in considerably shorter running time.

2.1 Mixed Integer Linear Programming

Linear programming was first proposed in 1947 by Dantzig (see Dantzig and Thapa [1997]) as a method to solve complex optimization problems. Dantzig states in his book (1997), that "Mathematical programming (or optimization theory) is that branch of mathematics dealing with techniques for maximizing or minimizing an objective function subject to linear, non-linear, and integer constraints on variables" (Dantzig and Thapa [1997], p. 1). The word programming here is a synonym for "planning" (Jensen and Bard [2003]). Hence, it is a method to formalize a planning problem mathematically, and then solve the mathematical model using algorithms tailored to the specific problem structure.

The terminology used in linear programming is as follows (Jensen and Bard [2003], p. 24):

- **Decision Variables** Algebraic variables, typically denoted as x. An assignment of a specific value to all variables is denoted a *solution* to the linear program. A solution might be restricted to integer or even binary value for some decision variables, leading to an integer linear program. In this thesis, we mainly deal with (mixed) integer linear programs.

- **Objective Function** An objective is a quantitative criterion, such as profit, costs or utility. An objective needs to contain at least one variable, whose value determines the size of the objective. The objective function can either be maximized or minimized.

- **Constraints** Constraints are equality or inequality restrictions on decisions. For example, a typical restriction in production planning is limited resources or for staffing decisions, there might be constraints determined by labor law. In a linear program, *all* constraints must be of a linear form, otherwise, we are dealing with a non-linear problem.

A simple example of an integer linear program is (see Jensen and Bard [2003]):

$$\text{Minimize} \quad z = \sum_{j=1}^{n} c_j x_j \tag{2.1.1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} \cdot x_j \geq b_i \qquad \forall i = 1, ..., m \tag{2.1.2}$$

$$x_j \in \mathbb{N} \qquad \forall j = 1, ..., n \tag{2.1.3}$$

A possible interpretation for the above program might be, that we want to minimize some costs associated with an assignment x_j, where c_j is some cost coefficient (2.1.1). Constraints (2.1.2) might indicate some minimum required reward b_i (right-hand side of the inequality), while the left-hand side sums up the actual assignments multiplied with some technological coefficient, for example. Here, x_j can only take positive integer values (Constraints (2.1.3)), leading to an integer program.

Multiple tools exist to solve (mixed) integer linear programs that have been formulated mathematically. The most prominent commercial tools are CPLEX (the current version we used throughout this thesis is 12.7) and Gurobi (currently in version 7.0.2), both of which apply variants of the well-known simplex algorithm to solve linear programs. As we had access to both mentioned commercial linear programming solvers through an academic license, CPLEX and Gurobi were the tools of our choice when it came to exact solution approaches of the mathematical modeling formulation of the health care scheduling optimization problems dealt with in this thesis. However, as we will show in the upcoming chapters, mixed integer linear programming solvers fail to find good quality (or even feasible) solutions in reasonable running time for large scale, real-world optimization problems. Hence, we will further focus on heuristic search concepts tailored to the specific planning problems.

2.2 (Meta)Heuristic Search Concepts

The algorithms applied throughout this thesis include Genetic Algorithms (GA), Variable Neighborhood Descent (VND) and Iterated Local Search (ILS). The search concepts will be shortly described in the upcoming sections:

2.2.1 Genetic Algorithms

2.2.1.1 Basic Scheme

Genetic Algorithms (GAs) are inspired by the biological evolution of species. The key idea is, that within a population of individuals, new individuals appear through recombination of existing ones (García-Martínez et al. [2018], Reeves [2010], Syswerda [1996], Zäpfel et al. [2010]). According to Darwin, natural selection will result in the fittest individuals having better chances of surviving and breeding, and less adapted individuals may not survive throughout the evolution process. Algorithm 1 gives an overview of the basic GA scheme: Starting from an initial population, $\mathcal{P}_0$, multiple succeeding populations $\mathcal{P}_{i+1}$ are formed. The next generation $i+1$ is built by best individuals of the previous population (elitism) and a recombined version (crossover) of selected individuals from $\mathcal{P}_i$. As in nature, some individuals are subject to mutation of their genes. The best found solution of the current population is then compared with the "fittest", i.e., best-known individual so far.

2.2.1.2 Design Decisions

Solution Representation In GAs, a solution – or individual – is represented by a chromosome, which typically are simple binary or permutation vectors describing a solution (Syswerda [1996]). Each possible chromosome must then be a legal solution to the problem. In scheduling problems, the solution "encoding" then needs to be "decoded" to receive the final schedule, this is sometimes also called "chromosome interpretation".

Initial Population The size of the (initial) population is an optimization parameter one must choose wisely. García-Martínez et al. [2018] mention three possibilities of building an initial GA population: Random solution, diverse solutions and heuristic solutions. Diverse solutions bias random solutions such that the diversity in the initial population is advanced, which is beneficial in searching for optimal solutions using GAs. Heuristic initial solutions will most likely lead to a better average objective in the initial population than pure random techniques.

Algorithm 1: Genetic Algorithm, Basic Scheme (e.g., García-Martínez et al. [2018], Reeves [2010])

```
1  P_0 ← CreateInitialPopulation();
2  s_best ← argmin_{p∈P_0}(ObjVal(p));
3  i ← 0;
4  repeat
5      P_{i+1} ← GetElites(P_i);
6      while |P_{i+1}| < |P_i| do
7          p_1 ← PerformSelection(P_i);
8          p_2 ← PerformSelection(P_i);
9          c ← Crossover(p_1,p_2);
10         c ← Mutate(c);
11         P_{i+1} ← P_{i+1} ∪ {c};
12     end
13     if argmin_{p∈P_{i+1}}(ObjVal(p)) < ObjVal(s_best) then
14         s_best ← argmin_{p∈P_{i+1}}(ObjVal(p));
15     end
16     i ← i + 1;
17 until termination criterion met;
```

Throughout this thesis, we generate the initial GA population by means of a combination of heuristic and random approaches.

Elitism When replacing a whole generation by a new one ("generational replacement"), the risk of dismissing good quality solutions during the search is high. Therefore, most GAs follow a concept called *elitism* (Zäpfel et al. [2010]). Here, a given number n of best individuals of a population "survive" until the next generation. We follow this strategy when applying genetic algorithms throughout this thesis.

Selection Generally, selection of individuals should be related to their fitness value. Well-known methods to incorporate the quality of the solution into the selection process are (Reeves [2010]):

(a) **Roulette Wheel Selection** uses a selection probability per individual that is proportional to its fitness. In the case of minimization, a transformation is required.

(b) In **Rank Selection** the individuals of a given population are ranked according to their fitness value, where the best out of N individuals gets rank N and the

worst individual gets rank 1. The selection probability is then proportional to the individual's rank.

(c) **Tournament Selection**: Here, a set of individuals is chosen randomly, among which the best one is selected. The size of the set is a parameter which needs to be optimized.

We will later in this thesis concentrate on the rank selection operator, as preliminary tests have shown its superiority in the setting of radiotherapy appointment scheduling.

Crossover & Mutation The type of crossover and mutation operators used within a GA is conditional to the solution representation on the one hand and the constraints on the solution representation on the other. Werner [2013] gives a broad overview of frequently used crossover and mutation operators both in the case of a binary and a permutation representation. In Chapter 4, we will introduce crossover (and mutation) operators that are tailored to the problem at hand.

Termination As mentioned in Reeves [2010]: "unlike simple neighborhood search methods that terminate when a local optimum is reached, GAs are stochastic search methods that could in principle run forever". Hence, a termination criterion is needed. Typically, a genetic algorithm could end after a given number of iterations or when a population's diversity falls below a given threshold (Reeves [2010]). This thesis, however, focuses on the comparison of various search concepts and therefore our termination criterion is a simple time limit (as in seconds).

2.2.1.3 Variants and Extensions

One major drawback of genetic algorithms is that the diversity of the population might diminish resulting in a rather homogeneous population. This might cause the algorithm to be stuck in a local minimum early during the optimization – also-called "premature convergence". To decelerate this effect, Affenzeller and Wagner [2004] proposed a variant of the basic genetic algorithm scheme, in which they "control genetic drift within the population by advantageous self-adaptive selection pressure steering".

Algorithm 2 gives an overview of this variant, which – as we will show later throughout this thesis – is beneficial also in our problem setting. The main difference to the basic GA scheme is, that we here force part of the new population to be formed by children outperforming their parents Affenzeller et al. [2009], as a new individual is only selected for the next generation if it is at least as fit as its worse parent (reflected by a comparison factor of 0.0; a larger comparison factor would

result in an even stricter criterion on the fitness of the children). Typically, one aims at building a high percentage (in Algorithm 2, 70%) of the next generation with such "good" children. However, the number of reproductive steps to achieve this goal is limited (in Algorithm 2 to five times the population size) in each iteration. If one has reached either the minimum required percentage of "good" children or the maximum number of reproductive steps, the rest of the next generation is filled up with "bad" children. Hence, the algorithm contains two sets of solutions: $\mathcal{P}_t$ forms the current population during iteration t and C^B contains children that did not reach the success criterion of being better than at least their worst parent and therefore do not immediately contribute to the next generation, where the latter group is only potentially used to fill up the population.

Algorithm 2: Genetic Algorithm with Offspring Selection (Affenzeller and Wagner [2004])

$\mathcal{P}_0 \leftarrow$ CreateInitialPopulation();
$s_{best} \leftarrow \text{argmin}_{p \in \mathcal{P}_0}(\texttt{ObjVal}(p))$;
$i \leftarrow 0$;
repeat
  $\mathcal{P}_{i+1} \leftarrow$ GetElites($\mathcal{P}_i$);
  $C^B \leftarrow \emptyset$;
  while $|\mathcal{P}_{i+1}| < 0.7 \cdot |\mathcal{P}_i| \wedge |\mathcal{P}_{i+1}| + |C^B| < 5 \cdot |\mathcal{P}_i|$ **do**
    $p_1 \leftarrow$ PerformSelection($\mathcal{P}_i$);
    $p_2 \leftarrow$ PerformSelection($\mathcal{P}_i$);
    $c \leftarrow$ Crossover(p_1,p_2);
    $c \leftarrow$ Mutate(c);
    if *ObjVal*(c) < *min*(*ObjVal*(p_1),*ObjVal*(p_2)) **then**
      $\mathcal{P}_{i+1} \leftarrow \mathcal{P}_{i+1} \cup \{c\}$;
    else
      $C^B \leftarrow C^B \cup \{c\}$;
    end
  end
  while $|\mathcal{P}_{i+1}| < |\mathcal{P}_i|$ **do**
    $c \leftarrow$ ChooseRandomElement(C^B);
    $\mathcal{P}_{i+1} \leftarrow \mathcal{P}_{i+1} \cup \{c\}$;
    $C^B \leftarrow C^B \setminus \{c\}$;
  end
  if $\text{argmin}_{p \in \mathcal{P}_{i+1}}(\texttt{ObjVal}(p)) < \texttt{ObjVal}(s_{best})$ **then**
    $s_{best} \leftarrow \text{argmin}_{p \in \mathcal{P}_{i+1}}(\texttt{ObjVal}(p))$
  end
  $i \leftarrow i + 1$;
until *termination criterion met*;

2.2.2 Variable Neighborhood Descent, Local Search

2.2.2.1 Basic Scheme

Variable Neighborhood Descent (VND) is a local search routine that searches different neighborhood structures for improvements of the current solution, ultimately reaching a local minimum at the end of a VND run (Duarte et al. [2018]). Once more, the solution representation – as in the GA setting – plays an important role when defining neighborhoods and neighbors. However, the benefit of VND is to search for better solutions than the current best found solution in multiple, i.e., k_{max} neighborhoods. A final solution is then a local minimum with regard to all k_{max} neighborhoods.

Algorithm 3 gives an overview of the basic VND version with "best improvement" strategy. The algorithm starts from a single starting solution s, and consequently the best found solution of the VND, s^* equals s in the beginning. k defines the neighborhood index. We repeatedly search the k-th neighborhood for the best neighboring solution, s' (line 4 in Alg. 3). In case the objective value of s' improves on the previous best objective (s^*), we accept the current solution as the new best and continue searching from this solution in the first neighborhood ($k \leftarrow 1$). If we did not find any improving neighboring solution, we continue the search within the next neighborhood ($k \leftarrow k + 1$), until we reach the final neighborhood ($k = k_{max}$).

Algorithm 3: Variable Neighborhood Descent, Best Improvement (Hansen et al. [2010, 2008])

```
   Input : s
1  k ← 1;
2  s* ← s;
3  repeat
4  |  s' ← argmin_{y∈N_k(s*)} ObjVal(y);
5  |  if ObjVal(s') < ObjVal(s*) then
6  |  |  s* ← s';
7  |  |  k ← 1;
8  |  end
9  |  else
10 |  |  k ← k + 1;
11 |  end
12 until k = k_max;
13 return s*;
```

As an alternative to the best improvement strategy, "first improvement" is commonly applied in VNDs. Here, instead of evaluating every single neighboring solution and consequently s' being the best found neighbor, we accept the first found neighboring solution s' for which $\texttt{ObjVal}(s') < \texttt{ObjVal}(s^*)$ as our new best solution. This speeds up the search per iteration but might lead to smaller improvement steps. Hence, for each problem individually, preliminary tests must be done to find the strategy to be favored. We will show a case in Chapter 4, were best improvement was more beneficial than first improvement. The opposite is true for the staff scheduling problem presented in Part II.

2.2.2.2 Design Decisions

Neighborhoods The types of neighborhoods searched for better solutions again are highly problem-related and will be presented in more detail in the upcoming chapters. Another important aspect is the determine the optimal number of neighborhoods k_{max}. All chosen neighborhoods should contribute to the search for better solutions (see Chapter 4 for further tests on the performance of the individual neighborhoods).

Neighborhood Size The size of a neighborhood and therefore the number of possible neighbors of a given starting solution might be vast and searching the complete neighborhood might therefore be time-consuming. Hence, it is common in practice to impose a limit on the number of evaluated neighbors in each iteration of the VND. As we are dealing with highly complex and rich real-world problems in this thesis, limiting the number of evaluated neighbors per iteration was necessary every time we applied a VND.

2.2.3 Iterated Local Search

2.2.3.1 Basic Scheme

Iterated Local Search (ILS) is a simple metaheuristic solution technique that iteratively calls an improvement method, with a modified starting solution in each iteration. The primary goal is to thereby run into several alternative local optima. We present the pseudo code of ILS – as in Lourenço et al. [2010] – in Algorithm 4. First, an initial solution is generated as a starting point to the algorithm. A local search approach is applied to this solution, such that the initial solution is instantly improved upon. Then, until a predefined termination condition is met, the current solution is slightly modified ("perturbed"), and the same local search procedure is started on the latter solution, leading to a local optimum. Finally, an acceptance criterion determines whether the currently found solution $s^{*\prime}$ replaces the incumbent solution for the next iteration.

Algorithm 4: ILS, see Lourenço et al. [2010]

```
1  s_0 ← GenerateInitialSolution;
2  s* ← LocalSearch(s_0);
3  s_best ← s*;
4  repeat
5  |   s' ← Pertubation(s*);
6  |   s*' ← LocalSearch(s');
7  |   if ObjVal(s*')<ObjVal(s_best) then
8  |   |   s_best ← s*'
9  |   end
10 |   s* ← AcceptanceCriterion(s*, s*');
11 until termination condition met;
```

2.2.3.2 Design Decisions

Initial Solution The quality of the initial solution s_0 can be important for the performance of the ILS, especially if the goal is to find a high-quality solution as quickly as possible as the local search procedure would then reach the local optimum requiring fewer iterations (Lourenço et al. [2010], Stützle and Ruiz [2018]). Most commonly, greedy construction heuristics lead to a reasonable starting solution, but also random starting solutions are standard choices in practice. In Chapter 4, we apply a more advanced strategy and build a family of greedy, randomized solutions, the best of which forms the starting point for the ILS.

Local Search Stützle and Ruiz [2018] mention, that actually any improvement method that "takes as input a complete candidate solution and returns a potentially improved candidate solution upon completion could be used" here. In our studies, we mainly apply VND as a local search procedure within ILS.

Perturbation Perturbation is required to escape from local optima during the search process. Every time the local search gets stuck in a local minimum, a perturbation of the current solution is performed, serving as a starting point for the next local search application (Stützle and Ruiz [2018]). The intensity of a perturbation highly affects the algorithm performance: A strong perturbation might behave like a random restart and part of the structure and quality of the previous incumbent might be lost. Perturbation that is too weak might result in the algorithm reaching the same local minimum once more (Lourenço et al. [2010]).

The perturbation strategy is highly problem specific. In Chapter 4, we apply two perturbation strategies iteratively: One with low perturbation strength and another one with higher strength. In Chapter 8 we compare a classic perturbation strategy with a random restart and a greedy restart approach. The latter two are no perturbation strategies as such but correspond to a modified version of the ILS, a (greedy or random) multi-start algorithm (Martí et al. [2018]).

Acceptance Criterion The acceptance criterion determines whether solution $s^{*\prime}$, found by the latest local search step, replaces the current incumbent solution s^*. Lourenço et al. [2010] describe two extremes of acceptance criteria during ILS. Either the new solution is accepted only if it improves s^* (i.e., "better" acceptance criterion), or it is accepted in any case (i.e., "random walk"). Between these extremes, many intermediate acceptance criteria can be found in prior literature, such as threshold-based or probabilistic ones (simulated annealing). Preliminary results have shown that in our case, the better criterion outperforms all other mentioned approaches.

Part I

Radiotherapy Patient Appointment Scheduling in an Ion Beam Facility

Chapter 3

Introduction to Radiotherapy Appointment Scheduling

3.1 General Problem Setting

The worldwide number of patients diagnosed with cancer has steadily increased over the past decade, from approximately 10 million cases (and 6 million deaths) in 2003 to around 14.1 million cases (8.2 million deaths) in 2012, reflecting an increase of 40% in only 10 years (Steward and Wild [2014]). Projections for 2030 range between 17.1 and 22.2 million cases, which equals an increase of 21.3% to 57.4% (see Bray et al. [2012], World Health Organization [2012]). Radiation therapy, or short radiotherapy, is commonly prescribed in addition to or instead of chemotherapy or surgery. The goal is to deliver a maximum amount of radiation to kill the cancer while sparing the healthy tissue surrounding the tumorous region (Washington and Leaver [2016]). In classical photon radiotherapy, which is available in virtually every hospital worldwide, radiation is delivered using a linear particle accelerator (linac) that supplies x-rays. More advanced but less numerous ion beam centers (only roughly 70 centers exist in the world, 48 of which have multiple treatment rooms, see PTCOG [2017]) use protons and/or carbon ions to achieve superior dose conformity and thereby lower the chances of patients developing secondary tumors later in life (Ohno [2013]). However, because these ion beam centers also use significantly larger particle accelerators, their operations are much more costly than is classical radiotherapy using linacs, and the efficient usage of the beam resource is crucial. Hence, throughout this part of the thesis, we focus on minimizing the idle time on this costly resource as our primary objective.

In this part we analyze and solve a real-world radiotherapy appointment scheduling problem arising in a recently opened, specialized ion beam center close to Vienna, Austria, called MedAustron. Ion beam facilities are typically equipped with one

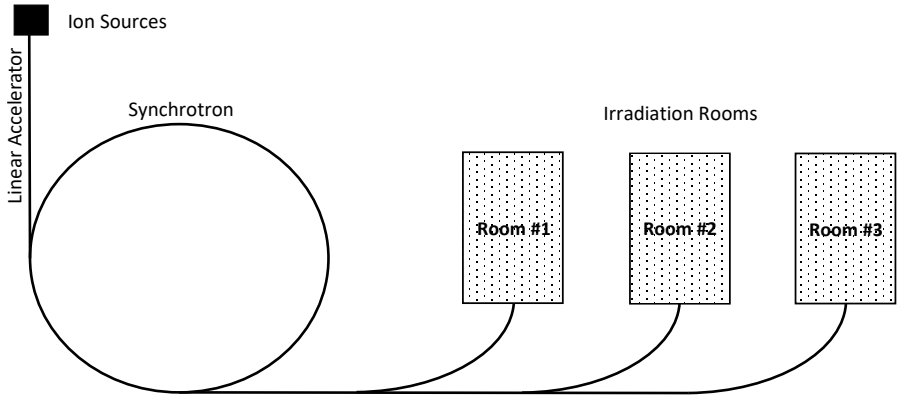


Figure 3.1: Facility Plan of MedAustron, Wiener Neustadt, Austria.

particle beam that serves multiple treatment rooms. The center features three different treatment rooms, equipped with (1) horizontally directed, (2) vertically and horizontally directed and (3) 180-degree rotatable particle beams, as shown in Figure 3.1. The particle beam – consisting of either protons or carbon ions – first moves through a linear accelerator, followed by multiple circulations through the synchrotron, where the beam gets accelerated to two-thirds of the speed of light, until it finally moves to one of the three treatment rooms. The beam can only serve one room at a time though, so we consider the particle beam the bottleneck resource in the irradiation process. Switching between the two offered particle types (protons and carbon ions) requires a set-up time of approximately 3 minutes.

MedAustron plans to treat approximately $P = 1000$ patients per year. A radiation treatment of one patient $p \in \mathcal{P}$ consists of both a pre-treatment phase and the irradiation phase – the actual treatment phase – itself. During the pre-treatment phase, multiple examinations take place, followed by intensive treatment planning, during which radio-oncologists (ROs), together with medical physicists, determine the dose of one treatment appointment (called a "fraction"), the beam direction (and therefore the required treatment room) and the number of recurring daily treatment appointments a patient should receive. On average, patients need to attend 20 daily treatments with an estimated average irradiation duration of 8 to 10 minutes, depending on the particle type. The planned irradiation duration is calculated during the treatment planning. However, as we will show in Chapter 5, this duration is highly stochastic. Additionally, during the treatment planning, medical physicists and doctors define the particle type used and develop a so-called immobilization device, which helps the patient lie still during the treatment.

The second phase, the actual treatment phase, consists of the previously fixed number of daily treatment activities (called DTs), imaging, and examination appointments, all of which require multiple resources simultaneously. Each DT

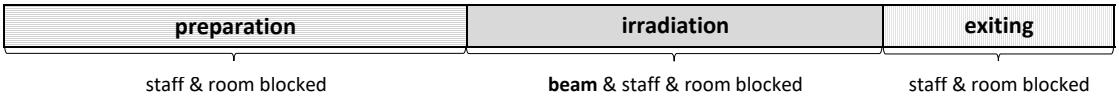


Figure 3.2: Phases of an Irradiation Appointment.

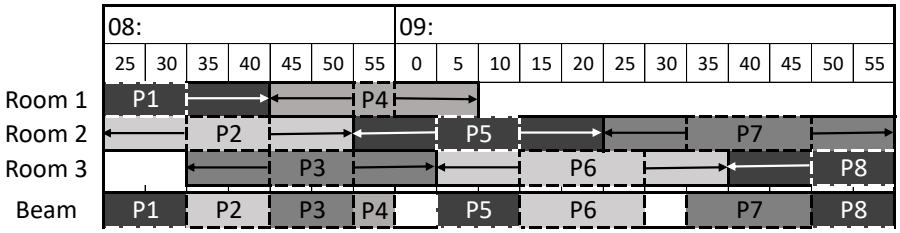


Figure 3.3: Example Schedule.

activity consists of three inseparable tasks: (1) a set-up time, or preparation activity, in which the patient is prepared for the treatment inside the treatment room, which means he/she is immobilized, fixed to the treatment bench and the correct positioning of the patient is assured through imaging; (2) the irradiation itself, which requires the use of both the beam resource and the treatment room; and (3) a tear-down time, or exiting activity, during which the patient is released from the treatment bench and finally the occupied treatment room is deallocated. The three phases are depicted in Figure 3.2. Scheduling two patients that require the same treatment room consecutively therefore causes extensive idle time on the beam resource (tear-down time of the first patient and set-up time of the second patient), so we prefer schedules that alternate between the treatment rooms. Figure 3.3 gives a first example of how a treatment schedule might look like.

Note, however, that the sum of the set-up and tear-down times equals an average of 18 minutes, whereas a treatment takes on average 8 to 12 minutes, depending on the particle type. Therefore, even alternating between two rooms might lead to beam idle time. An ideal schedule would interleave the three rooms, as depicted in Figure 3.3. Here, even though the proposed scheduling pattern applies in the beginning and the corresponding schedule is considerably tight, idle time on the beam resource between patient 4 and patient 5 is *unavoidable*, because patient 5 could not have started his set-up in room 2 earlier (the room was still blocked by the tear-down of patient 2). The same applies to the idle time between patients 6 and 7. This situation makes it considerably harder to calculate a reasonable lower bound for the minimum necessary idle time on our bottleneck resource (see Maschler et al. [2017b], whose work is inspired by the same real-world problem).

Throughout this part of the thesis, we use the term “beam” in two ways. First, it refers to an actual particle beam which supplies three unique treatment rooms, each at different angles. The beam can only serve one room at a time. Second, we

call our primary resource the "beam" in reference to the machine whose utilization we seek to optimize. This version of the beam is not only the primary resource but also the bottleneck resource in our problem setting.

The problem can be formalized as a complex job-shop scheduling problem with multiple custom constraints that need to be considered. Assigning treatments to days and scheduling the exact starting times of the activities to maximize facility usage and thereby minimize patients' waiting times before they start treatment is of utmost importance.

3.2 Overview of Part I

Chapter 4 deals with the deterministic, long-term scheduling of appointments in an ion beam radiotherapy facility. We introduce recurring optional appointments that, to the best of our knowledge, have not been considered in radiotherapy treatment scheduling before. Furthermore, we consider time windows between each activity for a patient that guarantee stable start times during the treatment phase. As solution techniques, we present three metaheuristics, namely, a genetic algorithm and an iterated local search, whose operators are tailored to the problem at hand, as well as a combination of these two approaches in a third algorithm. In doing so, we achieve a highly effective method to solve the problem of scheduling recurring radiotherapy appointments in the ion beam facility. The time horizon of real-world problem instances reflects various weeks and contains up to 10,000 activities to be scheduled.

Chapter 5 is dedicated to stochastic optimization of radiotherapy schedules, as various disruptions may occur during the execution of a given schedule (e.g., room unavailability for a longer period). We analyze data on past activity durations and fit theoretical distributions to the data. During stochastic optimization, we add patient waiting time to the objective function additional to the minimization of beam idle time.

Finally, Chapter 6 concludes Part I.

Chapter 4

Deterministic Radiotherapy Scheduling

This chapter is organized as follows: Section 4.1 presents the formal problem statement of the radiotherapy patient scheduling problem (RPSP) and discusses the constraints that arise at ion beam facilities in particular. Section 4.2 gives insight into related work on the radiotherapy patient scheduling problem. Section 4.3 is devoted to the mathematical programming formulation of the underlying scheduling problem. In Section 4.4, we discuss the three heuristical solution methods – two stand-alone methods and one hybrid algorithm – the results of which are in Section 4.5. Finally, Section 4.6 summarizes this chapter.

4.1 Problem Statement

4.1.1 General Constraints and Decisions

The treatment of one patient $p \in \mathcal{P}$ consists of a predefined number of recurring (almost) daily irradiation appointments (daily treatments, [DTs]) with fixed duration and resource requirements (e.g., treatment room, particle type, assigned RO). Some patients need to attend regular, additional imaging appointments (positron emission tomography [PET]) directly after the DTs to ensure the accuracy of the irradiation treatment. Between DTs, patients regularly (i.e., once within a span of five consecutive days during the treatment phase) see their assigned RO for a control examination (weekly control examination [WCE]). The sequence of these activities is fixed, as is shown by the "activity chain" in Figure 4.1, where "FDT" and "LDT" denote the first and last DT, respectively. Note, that both PET and WCE can take place after any DT (dashed boxes in Figure 4.1). These activities are optional in the sense that they must take place once within every span of five consecutive days,

Figure 4.1: Activity Chain

but the optimization algorithm must determine which WCE and PET activities should be scheduled and which ones can be skipped.

The FDT must be scheduled between the patient's specific release date and due date. As mentioned, the irradiation appointments then take place almost every day, and only one DT can take place per day. Although the total number of DTs, N_p^{DT}, is fixed by the RO, the days to which the treatments are assigned may vary slightly: starting from the FDT until the day of the LDT, patients need at least four irradiation treatments within every span of five consecutive days. The treatment phase then corresponds to the time period between the FDT and the LDT.

The treatment activities within the activity chain of one patient are tied together using minimum and maximum time-lags ("finish-start relations"). For example, to deliver accurate results, a PET appointment must start no later than 15 minutes after the preceding DT irradiation has finished. To maximize the patient's convenience, the DT activities also should take place at approximately the same time on every treatment day during a week. We therefore introduce a day time window for each week a patient receives treatment, which is defined by its mid-point. We call this mid-point the "stable starting time", as the approximate time a patient comes in for treatment on each day of the given week. Only deviations smaller than 30 minutes from the defined stable starting time are accepted, creating an even tighter time window for the DT activities. Furthermore, the stable starting times of two consecutive weeks may only differ by a maximum of 4 hours, allowing for changed approximate treatment times between weeks. Violations of the time windows formed by both finish-start relations and stable starting times result in penalties in the objective function.

Finally, some activities can be executed on alternative resource sets. For example, it might be preferable for the patient's assigned RO to perform the WCE, but if he or she is busy, any other RO on duty can undertake the examination. Each resource has a capacity of exactly one unit. Due to this unary resource capacity, the problem constitutes a complex job shop scheduling problem with custom constraints, alternative and preferred resource sets, recurring and partly optional activities, time windows and stable activity starting times.

Scheduling the patient appointment then consists of the following tasks:

- Assigning **treatment days** to patients, considering the release day and due day of the FDT and the minimum treatment pattern (four treatments in five

consecutive days).

- Defining which of the **optional activities** should take place and which ones can be skipped.

- Choosing which of the **alternative resources** should be used to operate the activities (that are taking place).

- Defining a **starting time** for all activities that need to be scheduled.

The necessary conditions that constrain the problem are listed below in words only. The corresponding mathematical modeling formulation is given in Section 4.3:

- Optional activities: During the treatment phase, starting from the FDT of a patient p until his LDT, at least one WCE (and one PET if required for the patient) need to be scheduled per week.

- Resources: Exactly one out the alternative resource sets needs to be chosen for the execution of the activity (that is taking place). Additionally, each resource can only perform one activity at a time (this also applies to the beam resource). All resources of the chosen resource set need to start the activity simultaneously and are occupied during the whole duration of the activity plus additional set-up and tear-down time if required.

- Activity sequencing: For each patient, the activity chain defines the sequence in which the activities need to take place. An activity needs to start within a fixed time window after the end of the preceding activity. A delayed start causes a penalty in the objective function (soft constraint).

- DT activities: For each patient, the predefined number of DTs needs to be scheduled. A minimum of four DTs must be scheduled within every span of five consecutive days, starting from the FDT.

- FDT start: For each patient, the FDT needs to be scheduled within the time window indicated by the patient specific release day and due day.

- Stable activity starting times: The starting times of DT activities of one patient may only vary from the patient's stable time of the current week by $\pm$ 30 minutes. Furthermore, the stable starting times of two consecutive weeks may only differ by a maximum of 240 minutes for a given patient.

4.1.2 Optimization Objective

We aim to minimize the total operation time of the beam resource which is used during the second task of the DT – the irradiation. The total operation time consists of the active time and the induced idle time as well as potential set-up time due to particle switches which cause set-up time on the beam resource, while the actual number of patients to be treated is determined by the doctors and is not part of the optimization. We thereby produce tight schedules which allow for the machine to be used for research in the field of medical physics and particle physics during the times when no patients are treated (typically at night). Simultaneously, we minimize penalties arising from the belated starting times of activities that violate either general time window constraints or the patient-specific stable treatment starting time per week. The mathematical formulation of the objective function can be found in Section 4.3.

4.2 Related Work

Appointment scheduling problems in health care systems arise in different environments, such as operating room scheduling and outpatient scheduling (e.g., Gupta and Denton [2008]). A literature review on multi-appointment scheduling problems, such as recurring treatment appointments (as in radiotherapy scheduling), is given in Marynissen and Demeulemeester [2019]. Radiotherapy scheduling in particular has attracted the interest of research groups during the past two decades. It first appeared in the literature in a review paper (Kapamara et al. [2008]), followed by basic algorithms for radiotherapy treatment booking proposed in Petrovic et al. [2006]. Various studies model this problem mathematically and solve it using standard Mixed Integer Programming (MIP) solvers (Conforti et al. [2008, 2010, 2011], Burke et al. [2011]). The different heuristics applied to the problem vary from pure constructive and hill climbing approaches (Kapamara and Petrovic [2009]) to greedy randomized adaptive search procedures (GRASP, Petrovic and Leite-Rocha [2008b]) and (multi-objective) genetic algorithm (GA) approaches (Petrovic et al. [2009, 2011]). Some authors focus on scheduling activities within the pre-treatment phase and use linear programming, simple dispatching rules, and GAs to solve this appointment scheduling problem (Petrovic and Castro [2011], Castro and Petrovic [2012]). In his PhD thesis, Leite-Rocha summarizes research on radiotherapy scheduling prior to 2011 and proposes various extensions to their mathematical models (Leite-Rocha [2011]).

More recent publications consider stochastic and dynamic attributes of the radiotherapy scheduling problem. Sauré et al. [2012] formulate the model as discounted infinite-horizon Markov decision process to identify good policies for

allocating capacity to incoming demand and thereby minimizing patient waiting times. Legrain et al. [2015] integrate stochastic patient arrival times into their model and develop a hybrid stochastic and on-line optimization algorithm. Gocgun [2018] also considers stochastic patient arrival times and introduces a simulation-based approximate dynamic programming approach to solve the problem. The PhD thesis by Men [2009] centers the analysis on the optimal mix of patients and diagnoses for an ion beam facility, to maximize the throughput of patients. Lately, Vieira et al. [2016] published a literature review on radiotherapy resource planning and treatment scheduling and categorized the papers according to their hierarchy level, methods used, the extent of implementation and the potential impact on the performance. They conclude that future research could incorporate specialized clinical pathways and additional devices such as PETs.

This mentioned research stream thus mainly focuses on scheduling treatments for "classical" photon therapies, for which each treatment room is equipped with a distinct linear accelerator. Sequencing patients per day and thereby scheduling exact starting times for all patients is less crucial in these settings, and accordingly, two main strategies for scheduling activities within the treatment phase dominate prior literature:

1. Assign treatments to days. This approach incorporates an average resource profile and does not schedule exact starting times on each day. Therefore, it requires a second step, namely, patient sequencing per day (Petrovic and Leite-Rocha [2008a], Men [2009], Conforti et al. [2010], Burke et al. [2011], Sauré et al. [2012]).

2. Split the day into time slots of predefined lengths (e.g., 15 or 30 minutes) and allocate the treatments to these time slots (i.e., "block scheduling", Conforti et al. [2008, 2011], Legrain et al. [2015]). This approach allows for the immediate consideration of stable activity starting times, but it also assumes that the treatment duration will be more or less equal to the length of the time slot, which is not the case in ion beam therapy, for which treatment durations vary substantially according to the diagnosis received by the patient.

This vast variation in treatment durations, as well as the bottleneck resource "particle beam" that is shared among various treatment rooms, necessitates scheduling exact ("to-the-minute") starting times at ion beam facilities. Maschler et al. [2016] propose a detailed scheduling approach using both a GRASP procedure and an iterated greedy approach, which incorporates interconnected day and time assignment phases. They extend the latter approach in a subsequent study and yield even better results on a mid-term planning horizon (Maschler et al. [2017a]). However, they do not incorporate optional activities, and they focus on scheduling the core irradiation appointments. Using a surrogate model to estimate the

Sets	
Notation	Description
$\mathcal{P}$	Set of all patients, index $p \in \{1, ..., P\}$.
$\mathcal{O}_p$	Set of operations/activities of patient p.
$\mathcal{R}$	General set of resources, index $r \in \{1, ..., R\}$.
$\mathcal{K}_{pi}$	Set of resource requirements for patient p's activity i.
$\mathcal{R}_{pik}$	Set of eligible resources for activity i and patient p, and resource requirement k. If only one (compulsory) resource is available, then $\lvert\mathcal{R}_{pik}\rvert = 1$.
$\mathcal{D}$	Set of days in the planning horizon, index $d \in \{1, ..., D\}$.
$\mathcal{W}$	Set of weeks in the planning horizon,index $w \in \{1, ..., W\}$.
Φ_p	Set of activities belonging to the subgroup of daily treatment activities for patient p.
Ψ_p	Set of activities belonging to the subgroup of weekly control examination activities for patient p.
Ξ_p	Set of activities belonging to the subgroup of PET activities for patient p.

Table 4.1: Sets of the Mathematical Modeling Formulation

lower bound for the time the beam resource is required if the patients treated on the specific day are known (i.e., the day assignment phase has already finished), Maschler et al. [2017b] also apply this information iteratively to optimize the day assignment.

4.3 Problem Formulation

The objective function and constraints described in Section 4.1 can be formulated mathematically. Table 4.1 lists the symbols and sets used in the formulation of the problem; Table 4.2 summarizes all necessary input information; and Table 4.3 gives an overview of the decision variables of the mathematical modeling formulation.

4.3.1 Objective Function

The objective function minimizes the operation time and thereby the idle time of the beam while simultaneously minimizing penalties arising from time window violations of activity i of patient p and stable time violations for patient p and day d. Here, f_d denotes the operation time of the beam resource on day d, $\widetilde{\gamma}_{pd}$ describes penalty caused by violations of the stable starting time (see Eq. 4.3.19) and $\hat{\gamma}_{pi}$

Input Parameters	
Notation	Description
u_{piqjr}	Set-up time between activity i for patient p and succeeding activity j (patient q) on resource r.
v_{pir}	Set-up time of activity i for patient p on resource r.
w_{pir}	Tear-down time of activity i for patient p on resource r.
t_{pi}	Processing time of activity i of patient p on all resources.
$FS^{min}_{pi,p(i+\delta)}$	Minimum time from finish of activity i to start of activity $i+\delta$.
$FS^{max}_{pi,p(i+\delta)}$	Maximum time from finish of activity i to start of activity $i+\delta$.
$\underline{dw_d}$, $\overline{dw_d}$	Begin and end of day-time window of day d.
α'	Maximum intra-week variation from the stable starting time.
α''	Maximum inter-week variation from the stable starting time.
r_p	Release day for patient p's FDT.
d_p	Due day for patient p's FDT.
M	Large number.

Table 4.2: Input Parameters to the Mathematical Modeling Formulation

accounts for general time window violations (see Eq. 4.3.11). All three parts of the objective function are measured in time units (minutes), i.e., one minute of extra beam duration accounts for one minute of delay for the patient (either time window or stable time violation). As will be shown in Table 4.11 in Section 4.5.5, the latter two parts of the objective function tend to almost disappear during the optimization process.

$$\text{minimize} \sum_{d\in\mathcal{D}} f_d + \sum_{d\in\mathcal{D}}\sum_{p\in\mathcal{P}} \widetilde{\gamma}_{pd} + \sum_{p\in\mathcal{P}}\sum_{i\in\mathcal{O}_p} \hat{\gamma}_{pi}. \tag{4.3.1}$$

The model's constraints can be categorized into various subsections: resource constraints, sequencing and optional activities, linking constraints, recurring activities, stable activity starting times, and general treatment start time constraints.

4.3.2 Resource Constraints

Each activity i requires K_{pi} resources simultaneously for every patient p. For some resource requirements, there also exist multiple alternative resources r, defined by the set of eligible resources for resource requirement k, namely, $\mathcal{R}_{pik}$:

$$\sum_{r\in\mathcal{R}_{pik}} h_{pikr} = o_{pi} \quad \forall p\in\mathcal{P}, i\in\mathcal{O}_p, k\in\mathcal{K}_{pi}. \tag{4.3.2}$$

Decision Variables	
Notation	Description
h_{pikr}	Binary variable, set to 1 if activity i of patient p is assigned to resource r for resource requirement k.
s_{pir}	Starting time of activity i for patient p on resource r.
$\bar{s}_{pi}$	Starting time of activity i for patient p.
y_{piqjr}	Binary variable for immediate successor of activity i for patient p (namely patient q's activity j) on resource r.
o_{pi}	Binary variable, indicating whether activity i for patient p takes place or not.
π_{pid}	Binary variable, indicating whether activity i takes place on day d or not (=1 if $\underline{dw_d} \leq \bar{s}_{pi} \leq \overline{dw_d}$)
$\hat{s}_{pd}$	Starting time of treatment for patient p on day d (independent of the activity i, time between the start of day-time window and the scheduled starting time).
$\widetilde{s}_{pw}$	Stable starting time of treatment for patient p in week w.
$\bar{x}_{pd}$	Binary variable, indicating whether day d is within the treatment phase of patient p.
$\bar{z}_{pw}$	Binary variable, indicating whether week w contains the treatment phase of patient p.
f_d	Finish time of last activity on the beam resource on day d.
$\widetilde{\gamma}_{pd}$	Stable time violation on day d for patient p.
$\hat{\gamma}_{pi}$	Time window violation for patient p's activity i.

Table 4.3: Variables of the Mathematical Modeling Formulation

Constraints (4.3.2) assure that for each required resource, one of the eligible resources is chosen if the activity takes place (i.e., variable $o_{pi} = 1$). Then,

$$s_{pir} \leq h_{pikr} \cdot M \quad \forall p \in \mathcal{P}, i \in \mathcal{O}_p, k \in \mathcal{K}_{pi}, r \in \mathcal{R}_{pik}. \tag{4.3.3}$$

$$\sum_{r \in \mathcal{R}_{pik}} s_{pir} = \bar{s}_{pi} \quad \forall p \in \mathcal{P}, i \in \mathcal{O}_p, k \in \mathcal{K}_{pi}. \tag{4.3.4}$$

Constraints (4.3.3) fix the starting times of non-chosen resources to 0. Constraints (4.3.4) assign the exact same starting times $\bar{s}_{pi}$ to all resources chosen for activity i of patient p, as the sum over all starting times on all eligible resources has only one positive entry per resource requirement k due to the previous constraint. In turn,

$$\sum_{\substack{i' \in \mathcal{O}_p \\ i' > i}} y_{pipi'r} + \sum_{q \in \mathcal{P} \setminus \{p\}} \sum_{j \in \mathcal{O}_q} y_{piqjr} = h_{pikr} \quad \forall p \in \mathcal{P}, i \in \mathcal{O}_p, k \in \mathcal{K}_{pi}, r \in \mathcal{R}_{pik}. \tag{4.3.5}$$

$$\sum_{\substack{j' \in \mathcal{O}_q \\ j' < j}} y_{qj'qjr} + \sum_{p \in \mathcal{P} \setminus \{q\}} \sum_{i \in \mathcal{O}_p} y_{piqjr} = h_{qjkr} \quad \forall q \in \mathcal{P}, j \in \mathcal{O}_q, k \in \mathcal{K}_{qj}, r \in \mathcal{R}_{qjk}. \tag{4.3.6}$$

Constraints (4.3.5) and (4.3.6) thus give the immediate successor structure of activities on resource r. Finally,

$$s_{pir} + t_{pi} + u_{piqjr} \cdot y_{piqjr} - (1 - y_{piqjr}) \cdot M \leq s_{qjr} \quad \forall p \in \mathcal{P}, i \in \mathcal{O}_p, q \in \mathcal{P}, j \in \mathcal{O}_q, r \in \mathcal{R}. \tag{4.3.7}$$

$$s_{pir} + t_{pi} + w_{pir} \cdot y_{piqjr} + v_{qjr} \cdot y_{piqjr} - (1 - y_{piqjr}) \cdot M \leq s_{qjr} \quad \forall p \in \mathcal{P}, i \in \mathcal{O}_p, q \in \mathcal{P}, j \in \mathcal{O}_q, r \in \mathcal{R}. \tag{4.3.8}$$

Constraints (4.3.7) and (4.3.8) confirm that both the sequence-dependent set-up times and the non-sequence-dependent set-up and tear-down times are respected.

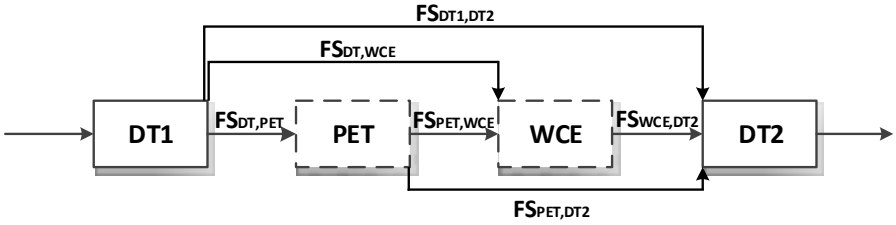


Figure 4.2: Finish-Start Relations among Optional Activities.

4.3.3 Sequencing and Optional Activities

Each activity i of patient p is associated with a binary variable o_{pi} that indicates whether the activity takes place or not. It is essential for optional activities (PETs and WCEs); and for DTs, this variable must equal 1. The starting times of activities that do not take place is then set to 0, using Constraints (4.3.9):

$$\bar{s}_{pi} \leq o_{pi} \cdot M \quad \forall p \in \mathcal{P}, i \in \mathcal{O}_p. \tag{4.3.9}$$

The problem of scheduling the activities of patient p considering the precedence relations (finish-start, FS) is further complicated by the optional activities within the activity chain, as shown in Figure 4.2. If, for example, a PET does not take place, the DT and WCE need to be tied together using the corresponding FS. However, if the PET is scheduled, the link between the DT and the WCE should be deactivated. In Constraints (4.3.10) and (4.3.11), successive activities that actually take place are connected using the minimum and maximum time lags between them. Constraints (4.3.11) further quantify the time window violation (belated scheduling) that is penalized within the objective function. Furthermore, $\Delta^{\max}$ denotes the maximum number of consecutive optional activities, in our case, $\Delta^{\max} = 2$.

$$\begin{aligned} \bar{s}_{p(i+\delta)} \geq \bar{s}_{pi} + t_{pi} + FS^{min}_{pi,p(i+\delta)} - \sum_{j=i}^{j=i+\delta} (M \cdot (1 - o_{pj})) \\ \forall p \in \mathcal{P}, i \in \mathcal{O}_p, \delta \in \{1, ..., \Delta^{\max} + 1\}. \end{aligned} \tag{4.3.10}$$

$$\begin{aligned} \bar{s}_{p(i+\delta)} \leq \bar{s}_{pi} + t_{pi} + FS^{max}_{pi,p(i+\delta)} + \sum_{j=i}^{j=i+\delta} (M \cdot (1 - o_{pj})) + \hat{\gamma}_{pi} \\ \forall p \in \mathcal{P}, i \in \mathcal{O}_p, \delta \in \{1, ..., \Delta^{\max} + 1\}. \end{aligned} \tag{4.3.11}$$

4.3.4 Linking Constraints

Constraints (4.3.12) to (4.3.15) serve as linking constraints. If an activity starts within a day-time window, the corresponding assignment variable π_{pid} equals 1. Constraints (4.3.13) also calculate the general daily starting time by subtracting the day start from the scheduled starting time. The treatment phase (days between FDT and LDT) of patient p can be calculated for both day d and week w using Constraints (4.3.14) and (4.3.15). Note, that we display these constraints as non-linear indicator constraints for better readability. The numeric tests in Section 5.6 use linearized versions of these constraints.

$$\pi_{pid} = 1 \iff \underline{dw_d} \leq \bar{s}_{pi} \leq \overline{dw_d} \quad \forall p \in \mathcal{P}, i \in \mathcal{O}_p, d \in \mathcal{D}. \tag{4.3.12}$$

$$\hat{s}_{pd} = \sum_{i \in \Phi_p} \bar{s}_{pi} \cdot \pi_{pid} - \underline{dw_d} \iff \sum_{i \in \Phi_p} \pi_{pid} = 1 \quad \forall p \in \mathcal{P}, d \in \mathcal{D}. \tag{4.3.13}$$

$$\bar{x}_{pd} = 1 \iff \sum_{0 \leq d' \leq d} \sum_{i \in \mathcal{O}_p} \pi_{pid'} \geq 1 \wedge \sum_{d \leq d' \leq D} \sum_{i \in \mathcal{O}_p} \pi_{pid} \geq 1 \quad \forall p \in \mathcal{P}, d \in \mathcal{D}. \tag{4.3.14}$$

$$\bar{z}_{pw} = 1 \iff \sum_{d=5 \cdot w-4}^{5 \cdot w} \bar{x}_{pd} \geq 1 \quad \forall p \in \mathcal{P}, w \in \mathcal{W}. \tag{4.3.15}$$

4.3.5 Recurring Activities

Constraints (4.3.16) ensure that at least four daily treatments are scheduled on five consecutive days within the treatment phase. Constraints (4.3.17) and (4.3.18) guarantee that at least one WCE (PET) is performed within five consecutive days during the treatment phase, respectively.

$$\sum_{d=s}^{s+4} \sum_{i \in \Phi_p} \pi_{pid} + M \cdot (1 - \bar{x}_{pd}) \geq 4 \quad \forall p \in \mathcal{P}, 1 \leq s \leq D - 4. \tag{4.3.16}$$

$$\sum_{d=s}^{s+4} \sum_{i \in \Psi_p} \pi_{pid} + M \cdot (1 - \bar{x}_{pd}) \geq 1 \quad \forall p \in \mathcal{P}, 1 \leq s \leq D - 4. \tag{4.3.17}$$

$$\sum_{d=s}^{s+4} \sum_{i \in \Xi_p} \pi_{pid} + M \cdot (1 - \bar{x}_{pd}) \geq 1 \quad \forall p \in \mathcal{P}, 1 \leq s \leq D - 4. \tag{4.3.18}$$

4.3.6 Stable Activity Starting Times

Inequalities (4.3.19) and (4.3.20) constrain the daily starting time for each patient to the stable starting time of the corresponding week as well as the stable starting times of two consecutive weeks, respectively:

$$|\hat{s}_{pd} - \widetilde{s}_{pw}| - M \cdot (1 - \sum_{i \in \Phi_p} \pi_{pid}) \leq \alpha' + \widetilde{\gamma}_{pd} \quad \forall p \in \mathcal{P}, w \in \mathcal{W}, 5w - 4 \leq d \leq 5w. \tag{4.3.19}$$

$$|\widetilde{s}_{pw} - \widetilde{t}_{p(w+1)}| - M \cdot (1 - \bar{z}_{pw}) - M \cdot (1 - \bar{z}_{p(w+1)}) \leq \alpha'' \quad \forall p \in \mathcal{P}, w \in \mathcal{W}. \tag{4.3.20}$$

4.3.7 Treatment Starting Time

No treatments can be assigned to patient p prior to his/her release day r_p, so

$$\sum_{i \in \Phi_p} \sum_{d=0}^{r_p - 1} \pi_{pid} = 0 \quad \forall p \in \mathcal{P}. \tag{4.3.21}$$

However, there has to be at least one treatment scheduled for patient p between his release day and due day, so

$$\sum_{i \in \Phi_p} \sum_{d=r_p}^{d_p} \pi_{pid} \geq 1 \quad \forall p \in \mathcal{P}. \tag{4.3.22}$$

4.3.8 Active Time of Beam Resource

Finally, we calculate the active time of the beam according to Constraints (4.3.23):

$$f_d \geq \hat{s}_{pd} + \sum_{i \in \Phi_p} t_{pi} \cdot \pi_{pid} \quad \forall d \in \mathcal{D}, p \in \mathcal{P}. \tag{4.3.23}$$

4.4 Solution Methodology

In this section we present three metaheuristic approaches to the proposed radiotherapy scheduling problem, to tackle the problem efficiently (as we show in Section 4.5, solving the exact MIP formulation, even for small to medium-sized problem instances, is intractable). We compare two metaheuristic paradigms, namely a population-based GA approach with a trajectory based local search heuristic and combine the two to a simple hybrid algorithm. Both methods have successfully been applied to related radiotherapy scheduling problems (e.g., GAs have been used in Petrovic et al. [2009, 2011], while local search has been performed in Petrovic and Leite-Rocha [2008b] and Kapamara and Petrovic [2009]).

Section 4.4.1 describes how we map the solution attributes to a multi-encoded solution representation. Section 4.4.2 gives some insight into the complexity of the problem and the proposed encoding scheme. In Section 4.4.3, we describe a greedy randomized method for creating initial solutions to the problem, followed by a detailed description of the decoding algorithm that transforms the solution encoding into a schedule with exact starting times in Section 4.4.4. Finally, Sections 4.4.5 to 4.4.7 present the three distinct solution methods: a GA, an iterated local search method (ILS), and a combination of the two approaches (combined GA and ILS, briefly cGAILS).

4.4.1 Solution Representation

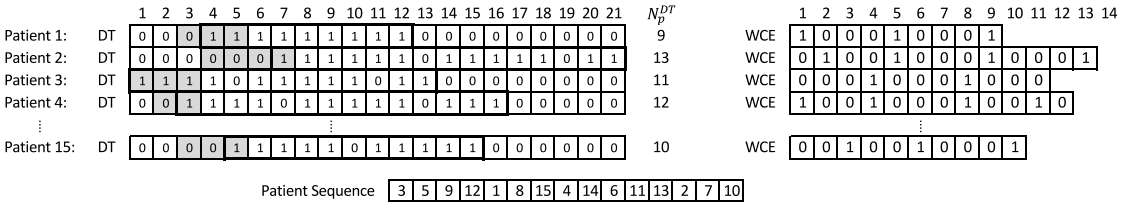


Figure 4.3: Solution Encoding, 15 Patients.

A solution to the RPSP is represented by a multi-encoded scheme which represents various specific characteristics of the solution. It consists of a set of three binary vectors of each patient $p \in \mathcal{P}$ and a list of patient indices. We distinguish three parts of the solution representation: The first part – a binary vector named the "DT assignment" – contains, for every patient $p \in \mathcal{P}$, an assignment of treatments to days. Days prior to the release time of the patient automatically remain unassigned. Between the FDT and the LDT (i.e., treatment phase), at least four treatments must be assigned to five consecutive days for the solution to be feasible. In the second part, two binary vectors for each patient $p \in \mathcal{P}$ indicate, when (i.e., after

which DT) a WCE and a PET is scheduled ("WCE/PET assignment"). An entry at the ith position in these vectors implies that a WCE (PET) has been added after the ith DT activity, so the sizes of those lists reflect the number of DTs to be scheduled for the focal patient; that is, the sizes vary from patient to patient. We must ensure a minimum of one WCE (PET) within each span of five consecutive days, resulting in at least one entry in the binary encoding over four consecutive DTs. Finally, the third part of the solution encoding consists of a vector of patient indices that displays the sequence in which the patients should be scheduled on a specific day ("patient sequence").

Figure 4.3 illustrates a solution representation of an RPSP for 15 patients ($P = 15$), where N_p^{DT} denotes the patient-specific number of required DTs. The bold frame within the DT assignment marks the treatment phase. The gray-shaded cells indicate the release and due dates for the FDT. Out of space considerations, we only display the WCE assignments; the PET assignments would reveal different allocations but the same sizes (N_p^{DT} of the patient p). In Figure 4.3, patient 1 starts his treatment on day 4. Directly after his first DT, a WCE is scheduled. The patient sequence lists patient 1 in the fifth position, so all predecessors will be scheduled prior to patient 1 on day d using the solution decoding algorithm (see Section 4.4.4).

4.4.2 Excursus on Problem Complexity

The presented solution representation already gives insight into the complexity of the underlying problem. The number of permutations of the DT assignment list for one patient p depends strongly on the number of DTs to be scheduled for this patient, namely N_p^{DT}, and can be calculated using Equation (4.4.1) (assuming the day of the FDT is fixed for patient p), with h_p^{max} denoting the maximum number of unassigned days allowed during the treatment phase. Given the above information, we define:

$$g(p) := \sum_{r=0}^{h_p^{max}} \binom{N_p^{DT} - 3 \cdot r + 2}{r}. \tag{4.4.1}$$

On average, a radiotherapy treatment consists of 20 DTs, which allows a maximum of 5 unassigned days during the treatment phase ($h_p^{max} = 5$) and a total of 657 feasible DT-to-day assignments, given the day of the FDT is fixed. A real-world instance would contain an average of 100 patients, each of which is assigned to an individual permutation list of DT assignments. Additionally, for each patient p, there exist on average four different "minimally occupied" WCE and PET assignments and a multiplicity of "non-minimally occupied" assignments. Finally, given P, or the number of patients to receive radiotherapy, $P!$ permutations of the

patient sequence exist. We then calculate the number of possible permutations by multiplying the three parts: $P! \times \prod_{p=1}^{P} g(p) \times 4^P \times 4^P$.

Therefore, instances including only five patients ($P = 5$) with 20 treatments each ($N_p^{DT} = 20$) and a fixed treatment start day already would imply $P! = 5! = 120$ permutations of the patient sequence, for each patient 657 feasible DT-to-day assignments, i.e., $\prod_{p=1}^{5} 657 = 1.22 \times 10^{14}$ possible assignments and both $4^5 = 1024$ possible WCE and PET assignments. This results in a total of 1.54×10^{22} solution permutations.

4.4.3 Initial Solutions

Initial, partly randomized solutions can be created using simple construction rules. We apply different strategies to the parts of the solution representation: For each patient, we first randomly fix the day of his or her FDT, noting the release and due days. Then the subsequent treatment days are fixed, with the premise that four treatments must be scheduled within five consecutive days. The probability of leaving one specific day d unscheduled (i.e., four prior days have a DT scheduled) is rather small, ranging between 0% and 30%. The same strategy applies for the WCE and PET assignments, where only one activity must be be assigned over four consecutive DTs.

Building the patient sequence requires a more sophisticated method though. We build a "global" patient sequence for all patients, in which we neglect the fact that some patients by definition will not be treated on the same day (because their release time will be later than other patient's LDT day, resulting in non-overlapping treatment phases). Using the assumption that all patients must be treated on a fictitious day d^*, we choose a random patient, whom we add as the first patient in the patient sequence. Then, we continuously add one more patient, who minimizes the total idle time of the beam, due to either room unavailability (i.e., the tear-down time of the previous patient is not yet finished when the set-up of the current patient should start) or set-up time due to particle type switches (from proton to carbon ion or vice versa). This strategy results in a starting solution where only in rare cases are two patients requiring the same treatment room scheduled successively.

4.4.4 Solution Decoding and Solution Evaluation

To decode the described solution representation into a schedule that provides exact starting times and resource decisions for each activity, we first transform the solution into a chronological (i.e., day-wise) prioritized activity list. We designed two decoding algorithms: The first (decoder 0, or chronological decoder) schedules activity starting times chronologically on a given day, according to the days and

patient sequences assigned in the chromosome. The second algorithm (decoder 1, or gap-filling decoder) permits deviations from the predefined patient sequence per day, in the case that availability gaps of the beam resource (i.e., idle times resulting from set-up and tear-down times in the rooms) might be reduced by inserting the current activity. We show in a preliminary study (which was published in Vogl et al. [2018a]) in Section 4.5.1, that decoder 1 is outperforming decoder 0, so we focus on the so-called "gap-filling decoder" throughout the second part of our computational study.

Algorithm 5 summarizes this gap-filling decoding algorithm, where we determine the exact starting time and the resource set on which the activity should be performed. We begin the search for a feasible starting time on the "preferred" resource set ($n = 1$, e.g., the assigned radio-oncologist for the WCE). We use function FindStartingTime$(i, n, \bar{s}_{pi}, l_{pi})$ to determine the earliest starting time for activity i on resource set n, with $\bar{s}_{pi}$ as the earliest time to start and as l_{pi} the latest possible starting time. We first search all N_{pi}-eligible resource sets for a *feasible* starting time (i.e., a smaller than or equal to l_{pi}). Hence, if no time feasible position is available on the preferred resource set, the second desired resource set is searched (e.g., any other radio-oncologist on duty), and so on. If no such starting time arises from any resource set, the first non-feasible (i.e., belated) starting time on the "preferred" resource set is accepted as the starting time for activity i, though it results in a penalty within the objective function.

Note, that function FindStartingTime$(i, n, \bar{s}_{pi}, l_{pi})$, which is depicted in Algorithm 6, does not necessarily add activities to resources chronologically, but allows activities to be scheduled to fill up "holes" in resources, such as might occur if we were to schedule two treatment activities in the same treatment room successively and therefore face idle time on the beam resource due to tear-down and set-up times. We then would aim to schedule activities requiring any other treatment room in between those activities to minimize the idle time on the beam resource.

The sequential nature of the decoding algorithm requires a post-scheduling evaluation of the treatment starting times to evaluate the stable time violations, because the stable time for each patient p and each week w needs to be assigned "globally" and not when scheduling the first activity of the week. Therefore, we solve a small linear program for each patient p after the schedule construction has finished to reveal stable time violations, as the stable time per week is a variable itself, not an input to the optimization. This small linear model contains only two variable types, namely, $\widetilde{s}_w$, the stable time of week w, and γ_d, the stable time violations on day d. The daily starting time $\hat{s}_d$ for the currently observed patient p, the day assignment variable π_d, and the weeks within the treatment phase $\mathcal{W}'$ have already been fixed during the solution decoding phase and are therefore input

Algorithm 5: Solution Decoding Algorithm

repeat
  Determine next activity i to be scheduled from the activity list;
  $n \leftarrow 1$;
  $\bar{s}_{pi} \leftarrow \infty$;
  $l_{pi} \leftarrow$ latest feasible starting time of activity i;
  while $\bar{s}_{pi} > l_{pi} \wedge n < N_{pi}$ **do**
    $\bar{s}_{pi} \leftarrow$ earliest feasible starting time of activity i;
    $\bar{s}_{pi} \leftarrow$ FindStartingTime$(i, n, \bar{s}_{pi}, l_{pi})$;
    $n \leftarrow n + 1$;
  end
  if $\bar{s}_{pi} > l_{pi}$ **then**
    $\bar{s}_{pi} \leftarrow$ FindStartingTime$(i, 1, l_{pi}, \infty)$;
  end
  Schedule activity i at the determined starting time $\bar{s}_{pi}$;
  Update time windows of the successive activities;
until *all activities scheduled*;

parameters. Accordingly, for each patient, we solve

$$\text{minimize} \sum_{d \in \mathcal{D}} \gamma_d \tag{4.4.2}$$

subject to:

$$|\hat{s}_d - \widetilde{s}_w| - \gamma_d - M \cdot (1 - \pi_d) \leq \alpha' \quad \forall w \in \mathcal{W}', 5w - 4 \leq d \leq 5w \tag{4.4.3}$$

$$|\widetilde{s}_w - \widetilde{s}_{w+1}| \leq \alpha'' \quad \forall w \in \mathcal{W}'. \tag{4.4.4}$$

The objective function is to minimize the sum of daily deviations from the stable starting times. These deviations are calculated using Constraints (4.4.3). Constraints (4.4.4) then assure that the inter-week stable time variation is smaller than the maximum allowed deviation, namely α''.

4.4.5 Genetic Algorithm

Genetic algorithms have been implemented successfully to solve the radiotherapy patient scheduling problem (Petrovic et al. [2009, 2011]). Hence, we decided to also apply a genetic algorithm to approach the radiotherapy appointment scheduling problem. In order to preserve feasibility during the evolutionary process, we introduce tailor-made and sophisticated crossover and mutation mechanisms.

Algorithm 6: FindStartingTime$(i, n, \bar{s}_{pi}, l_{pi})$

```
1  while s̄_pi < l_pi do
2      for all required resources r of resource set n do
3          check availability of resource r for the time period of the processing
           time t_pi including set-up time v_pir and tear-down time w_pir:
           [s̄_pi − v_pir, s̄_pi + t_pi + w_pir];
4          if resource unavailable then
5              s_r ← start of next idle time window of resource r;
6              if s̄_pi < s_r then
7                  s̄_pi ← s_r;
8              end
9          end
10     end
11 end
12 return s̄_pi;
```

Note, that in a preliminary study (see Section 4.5.1), we compared two genetic evolution strategies: In a classical genetic algorithm, one generation consists of n individuals, a small share of which survives until the next generation ("elitism"). The rest of the population consists of children created through crossover (and mutation) of two individuals of the parent population. The second strategy, offspring selection (OS), forces part of the new population to be formed by "good" children (Affenzeller et al. [2009]). A new individual is only selected for the next generation if it is at least as fit as its worse parent (reflected by a comparison factor of $= 0.0$). We aim at building 70% of the next generation with such "good" children. However, the number of reproductive steps to achieve this goal is limited to five times the population size in each iteration. As soon as we have reached either 70% "good" children or the maximum number of reproductive steps, we complete the rest of the child population with "bad" children. Hence, the algorithm contains two sets of solutions: $\mathcal{P}_t$ forms the current population during iteration t and C^B contains children that did not reach the success criterion of being better than at least their worst parent (see line 12 in Algorithm 2 on page 11) and therefore do not immediately contribute to the next generation. The latter solutions are potentially used if by producing five times the population's size as offspring is not sufficient to build the new population from successful children.

Algorithm 2 on page 11 gives an overview of the mentioned GA components. We will now describe all problem specific GA components in more detail in the following paragraphs:

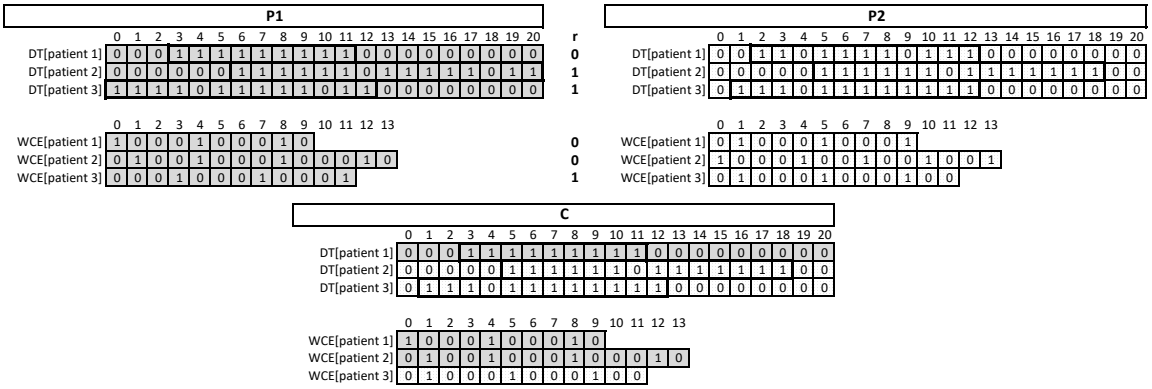


Figure 4.4: Patient-Wise Crossover.

Crossover(p_1,p_2) The multi-encoded solution representation demands specific crossover and mutation operators for the individuals to remain feasible throughout the evolutionary process. We use two crossover operators:

(1) The patient-wise crossover operator, where the whole DT assignment as well as the whole PET and WCE lists of one patient p is randomly inherited either by the first or the second parent. Figure 4.4 illustrates this crossover operator. Here, the child inherits the DT list for patient 1 from the first parent P1 (binary random variable $r = 0$), and r equals 1 for the DT lists of patients 2 and 3, resulting in the inheritance from the second parent's (P2) chromosome. This simple crossover only leads to limited variety in the binary encodings, because the combinations of the initial population dominate the search.

So, (2) we developed a tailor-made, day-wise crossover operator for the DT assignment part of the multi-encoded chromosome, illustrated in Figure 4.5. The day-wise crossover chronologically compares entries of the parents on a specific day d. If both parents have a DT assigned on day d, we naturally assign a treatment on this day (white entries). The same applies for the cases in which no treatment is planned on day d, observable by two "0" entries in the parent chromosomes. Gray-shaded cells indicate days with different allocations among the parents. The assignment of every yet undecided day d is fixed randomly, again defining the allocations chronologically. We first have to determine if leaving day d unassigned leads to an infeasible solution regarding the constraint of assigning four treatments within every five consecutive days. If so, we assign a treatment to this day in any case.

Otherwise, the probability of assigning a treatment to day d depends on the number of still missing DTs for patient p, $n_{missing}$, and the number of undecided days, $n_{undecided}$: $P(\mathrm{DT}_d) = \frac{n_{missing}}{n_{undecided}}$. We call this crossover strategy the "day-wise crossover operator."

The day-wise crossover and patient-wise crossover then are chosen randomly

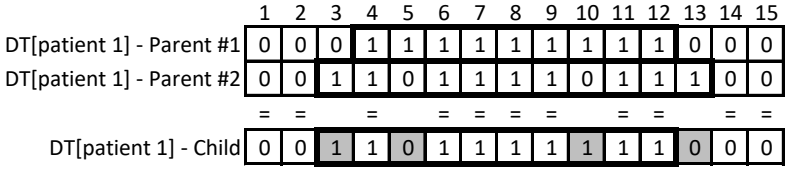


Figure 4.5: Day-wise Crossover.

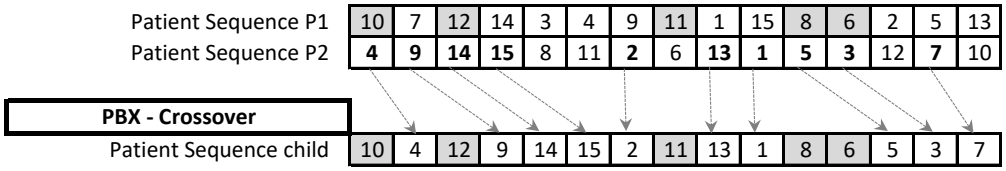


Figure 4.6: Position-based Crossover Operator.

during the genetic evolution of the DT assignments with equal probabilities. The WCE and PET assignments use only the patient-wise crossover operator. Finally, the patient sequence resembles a classic mutation encoding, so crossover and mutation operators known from genetic algorithm literature, such as the position-based crossover operator (PBX) can be applied (Syswerda [1996]). This crossover mechanism is displayed in Figure 4.6. Here, we choose positions randomly from the patient sequence of parent P1 and copy the entries to the offspring. The missing entries are then inserted to the yet empty positions according to the sequence in parent P2.

Mutate(c) To enhance the search, we apply mutation operators to 10% of the descendants in each generation. The DT assignment per patient is mutated by inverting the list within the treatment phase. The WCE and PET assignments are completely reversed, and the patient sequence is mutated using the well-known shift mutation operator, such that one random patient p is removed from the current sequence and reinserted at a random position within the sequence.

To choose individuals for reproduction (PERFORM SELECTION($\mathcal{P}_i$)), we employ the rank selection operator. In addition, 1% of the offspring population is composed of the best individuals of the parent population (GETELITES($\mathcal{P}_i$)).

4.4.6 Iterated Local Search

As an alternative approach to the GA, we introduce an ILS to solve the RPSP, where the local search step of the algorithm is formed by a variable neighborhood descent (VND). Algorithm 4 on page 14 gives an overview of the classic ILS components as in Lourenço et al. [2010], all of which are described in more detail in the following

paragraphs:

GenerateInitialSolution The initial solution of the ILS is defined by the best solution found within a pool of randomly generated solutions, which are constructed using a partly greedy, partly random approach described in Section 4.4.3. The pool size depends on the instance size with larger, real-world inspired instances in Section 4.5 having a pool of 200 initial solutions to choose from. Hence, one could imagine the initial solution to the ILS being the best solution among the initial GA population.

LocalSearch The local search part of the algorithm is formed by a VND, which operates on different parts of the solution representation in Section 4.4.1. Traditionally, the VND contains various different neighborhoods with increasing degree of disruptiveness, which allows the algorithm to leave potential local minima. Preliminary tests have shown that in our specific case, the following $k_d = 6$ neighborhoods contribute to the search for improvements to the solution fitness. (For details on these preliminary tests please refer to Section 4.5.3.)

1. Perform a random shift of a patient within the patient sequence (N1).

2. Invert the DT assignment of a random patient (N2).

3. Invert both the WCE and PET assignments of a random patient (N3).

4. Swap two random patients within the patient sequence (N4).

5. Rebuild the DT, WCE, and PET vectors for a random patient from scratch (random assignment, as in the starting solutions, (N5)).

6. Invert a random subsequence of a size between 2 and 5 of the patient sequence (N6).

If a better solution is found, the VND continues its search within the first neighborhood. In case no better solution can be found in any of the listed neighborhoods, we reach a local minimum, leading to the termination of the VND. Because the high complexity of the underlying problem leads to vast neighborhood sizes, we impose a limit on the number of evaluated neighbors in each iteration of the VND (see Section 4.5 for details). Experiments have shown that the best improvement policy during the neighborhood search of the VND is beneficial, which is why we follow this scheme.

Perturbation If the local search gets stuck in a local minimum, a perturbation of the current solution is performed, to leave the local minimum and restart the local search phase from another starting point. We use two different perturbation strategies, one less and one more invasive method. Every time a local minimum is found that improves the global best, we use the less invasive perturbation method. Otherwise, we alternate these approaches.

The less invasive method inverts the DT, WCE, and PET assignments of a random patient, and it also inverts a random part of the patient sequence of size $P/7$. The second perturbation strategy rebuilds a completely new assignment of DTs, WCEs, and PETs for a random patient (as in neighborhood 5 of the VND) and inverts a random part of the patient sequence of size $P/5$, with P denoting the total number of patients.

AcceptanceCriterion The acceptance criterion determines whether solution $s^{*\prime}$, found by the latest local search step, replaces the current incumbent solution s^* (see Algorithm 4 on page 14). Two extremes of acceptance criteria during ILS are described in Lourenço et al. [2010]. Either the new solution is accepted only if it improves s^* (i.e., "better" acceptance criterion), or it is accepted in any case (i.e., "random walk"). Between these extremes, many intermediate acceptance criteria can be found in prior literature, such as threshold-based or probabilistic ones (simulated annealing). Preliminary results have shown that in our case, the "better" criterion outperforms all other mentioned approaches.

4.4.7 Combined GA and ILS

Combinations of genetic algorithms and local search based algorithms, so-called memetic algorithms or genetic local search metaheuristics, have proven to be beneficial in the literature (see, e.g., Neri et al. [2012], who summarized work on memetic algorithms or Vela et al. [2010], who successfully interleaved a GA with a local search and called this hybrid form "genetic local search"). Because the power of ILS in our case highly depends on the quality of the initial solution, we investigated further into hybrids of the two presented approaches. However, classic memetic approaches that perform a local search on children in the GA were not competitive. Therefore, we present a simple but effective (see Section 4.5) combination of the two described stand-alone metaheuristics in a way that we first run the GA and afterwards, taking the best solution of the GA, we perform the proposed ILS approach. The available CPU time is distributed equally among the approaches.

4.5 Computational Results

The computational results presented in this section represent findings from the two deterministic radiotherapy appointment scheduling papers Vogl et al. [2018a,b]: Section 4.5.1 focuses on the comparison of the GA variants with and without offspring selection (OS) as well as the two decoding algorithms presented in Section 4.4.4 (see Vogl et al. [2018a]). The instance set used therein slightly differs from the one used in the later paper, which is thoroughly described in Section 4.5.2. We then discuss the preciseness of the neighborhoods within the iterated local search algorithm in Section 4.5.3.

We analyze small, toy instances to assess the performance of the metaheuristic solution approaches in comparison to the exact MIP formulation in Section 4.5.4. Finally, we discuss the computational results for large and medium-sized problem instances in Section 4.5.5 and compare the algorithms on various key figures associated with the solutions. Results of statistical tests to prove the dominance of one heuristic approach over the other are presented in Section 4.5.6. Sections 4.5.2 to 4.5.6 are published in Vogl et al. [2018b].

4.5.1 Preliminary Study I – Genetic Algorithm Variants and Decoding Algorithms

We test the proposed GA on medium and large, real-world-inspired problem instances, that slightly differ from the general experimental set-up used and described in the upcoming sections. The number of patients to be treated varies from 25 to 200. The number of treatments per patient is equally distributed between 8 and 12 for smaller instances (25 to 75 patients) and between 10 and 18 treatments for the larger ones (100 and 200 patients). All patients in the instances with 25 to 75 patients have release times and deadlines within the first week of the planning horizon, but we simulate staggered release times for the instances that include 100 and 200 patients. Therefore 25 patients start the treatment the first week, 25 patients start their treatment in the second week, and so on. The population size of the genetic algorithm is set to 100 for the 25 patients instance and to 200 for the larger instances. Accordingly, the optimization time limit increases with the instance size (see Table 4.4). The best 1% of the population survives until the next generation of the algorithm (elitism), and individuals are chosen for reproduction using the rank selection operator. Furthermore, 10% of the offspring are selected for mutation, using the operators described in Section 4.4.5.

Table 4.4 lists the averages of 16 replications for all five instances and the corresponding algorithmic settings (i.e., two decoding algorithms, with and without OS). The column "P" describes the number of patients to be scheduled, followed by the corresponding number of activities to be scheduled in the second column "Act".

With "LB", we denote a naive lower bound that consists of the sum of all activity durations that require the beam resource. This lower bound neglects unavoidable idle times of the beam, due to the set-up and tear-down times in the rooms as well as beam type switches. Therefore, it can give a first but still weak impression of the quality of the solution. In column "d", we compare the two mentioned decoding algorithms ($d = 0$ gives the pure chronological scheduling approach, $d = 1$ can fill gaps). As already mentioned, we compare the classical genetic algorithm (without OS) with a variant including offspring selection. The results of this comparison is in the columns denoted "Without OS" and "With OS." In the "Av. Fit." columns, we provide the average of best solution fitnesses after the time limit, followed by the average of the corresponding penalty terms in the columns denoted by "Av. Pen." The bold values represent the best found average solution fitness for the instance size.

Although the chronological decoding algorithm ($d = 0$) performs almost as well as the gap-filling decoding algorithm ($d = 1$) for the smaller instances, permitting the method to fill availability gaps is highly advantageous for larger instances. Employing offspring selection also is beneficial in our problem setting: The best found solution fitness significantly increases for almost all problem instances in both decoding algorithms ("OS impr."). p-values < 0.001 resulting from a two-sided t-test are marked with "***", p-values ≥ 0.05 are marked with "ns" (not significant). Finally, the "Gap" column indicates the difference between the best found solution for the instance and the naive lower bound.

Figure 4.7 depicts the evolutionary process of the 100 patients instance using the second decoding algorithm ($d = 1$) with the mentioned offspring selection procedure. Both the average population fitness and the best population fitness ameliorate rapidly during the first 100 generations and slightly flatten out after 200 iterations. Figure 4.8 compares the final fitnesses of the 16 replications after 7,200 seconds of running time with and without OS. The worst case performance with OS still outmatches the best case without OS. Furthermore, OS decreases the variation in solution fitness across the replications.

Hence, we decided to use the gap-filling decoder and the genetic algorithm with OS in all future runs of the algorithm.

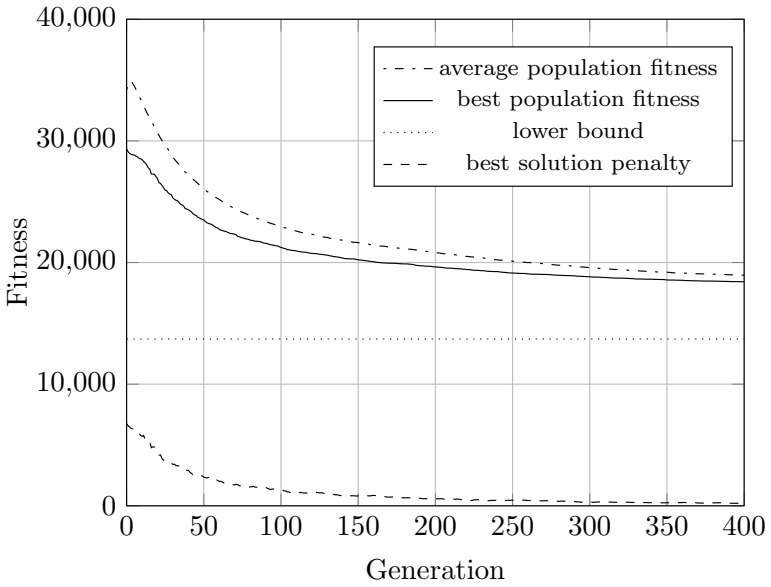


Figure 4.7: Genetic Algorithm Performance with OS, 100 Patients Instance, 16 Replications, $d = 1$.

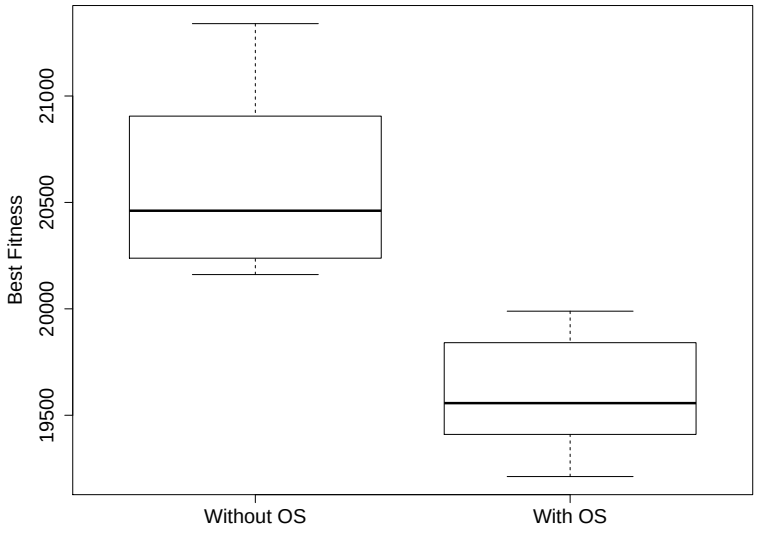


Figure 4.8: Comparison of Genetic Algorithm Performance with and without OS, 100 Patients Instance, 16 Replications, $d = 1$.

Table 4.4: Preliminary study – GA variants and decoding algorithms, real-world inspired instances, 16 replications per instance

					Without OS		With OS				
P	Act.	LB	d	Time	Av. Fit.	Av. Pen.	Av. Fit.	Av. Pen.	OS impr.	p	Gap
25	580	2,285	0	1,200	3,162.7	25.9	3,184.8	94.8	0.7%	ns	31.2%
			1	1,200	2,997.6	1.5	**2,997.1**	1.7	0.0%	ns	
50	1,209	4,972	0	2,400	7,756.6	304.1	7,213.7	463.7	-7.0%	***	30.0%
			1	2,400	6,778.4	47.5	**6,463.1**	15.5	-4.7%	***	
75	1,778	7,090	0	3,600	13,888.1	1,123.0	11,169.6	534.1	-19.6%	***	44.4%
			1	3,600	10,799.8	245.8	**10,234.8**	146.0	-5.2%	***	
100	3,392	13,717	0	7,200	26,089.9	1,467.1	22,387.5	788.3	-14.2%	***	43.0%
			1	7,200	20,585.5	907.1	**19,609.6**	556.4	-4.7%	***	
200	6,691	27,838	0	14,400	52,516.6	2,365.9	46,825.3	977.9	-10.8%	***	43.6%
			1	14,400	40,992.8	2,572.8	**39,978.4**	1,915.9	-2.5%	***	

4.5.2 General Experimental Set-up

The instances were generated using the knowledge of MedAustron staff about the underlying distribution functions. Patient-specific random variables came from the distributions summarized in Table 4.5. Small toy examples with 3 to 25 patients are created to assess the quality of the heuristic methods in comparison with exact solution methods; the larger, more realistic instances include 35 to 175 patients.

Additional instance settings are as follows:

$$u_{piqjr} = \begin{cases} 3, & \text{if } p \text{ and } q \text{ have different beam types and } r \text{ is the beam} \\ 0, & \text{otherwise.} \end{cases}$$

$$t_{pi} = \begin{cases} 30, & \text{if } i \in \Xi_p \\ 10, & \text{if } i \in \Psi_p \end{cases}$$

$$\alpha' = 30 \text{ minutes}$$

$$\alpha'' = 120 \text{ minutes}$$

$$FS^{min}_{pi,p(i+\delta)} = \begin{cases} 0, & \text{if } i \in \Phi_p \wedge (i+\delta) \in \Xi_p \\ 15, & \text{if } i \in \Phi_p \wedge (i+\delta) \in \Psi_p \\ 15, & \text{if } i \in \Xi_p \wedge (i+\delta) \in \Psi_p \end{cases}$$

$$FS^{max}_{pi,p(i+\delta)} = \begin{cases} 15, & \text{if } i \in \Phi_p \wedge (i+\delta) \in \Xi_p \\ 60, & \text{if } i \in \Phi_p \wedge (i+\delta) \in \Psi_p \\ 60, & \text{if } i \in \Xi_p \wedge (i+\delta) \in \Psi_p \end{cases}$$

For instances with $P \geq 35$, we acknowledge that several patients probably have already started their treatment, prior to the beginning of the planning horizon. According to our project partner, patients require an average of 20 treatments, resulting in the following scheme: For any day in the planning horizon, approximately 1/4 of patients will be having their first week of treatments; another quarter is within the second treatment week. Finally, 1/4 of patients has already had two treatment weeks and is currently in the third week of treatments and the last quarter of patients is finishing treatment after the current week. We follow this systematic approach when generating our real-world inspired problem instances. We therefore create 7 types of patients, as in Table 4.6. The first three categories have already started their treatment prior to the beginning of the planning horizon and have between 4 and 15 treatments left. Therefore, the release dates for these patients is 0. For 20% of these patients, we assume that they could have a day off on day 0, resulting in a due date of 1 instead of 0. In categories 4 to 7, patients

Attribute	Probabilities
Beam type	$P(BeamType_p = \text{Proton}) = 0.50$
	$P(BeamType_p = \text{Carbon Ion}) = 0.50$
Duration irradiation	$\sim \mathcal{N}(12,5) \iff BeamType_p = \text{Proton}$
$t_{pi} \quad \forall i \in \Phi_p$	$\sim \mathcal{N}(8,5) \iff BeamType_p = \text{Carbon Ion}$
Room	$P(Room = 1) = 0.33$
	$P(Room = 2) = 0.33$
	$P(Room = 3) = 0.33$
Duration in-room set-up	$P(v_{pir} = 12) = 0.80$
	$P(v_{pir} = 22) = 0.20$
Duration in-room teardown	$P(w_{pir} = 3) = 0.70$
	$P(w_{pir} = 6) = 0.30$
aRO	$P(aRO = 1) = P(aRO = 2) =$
	$P(aRO = 3) = P(aRO = 4) = 0.25$
PET	$P(PET = true) = 0.50$

Table 4.5: Probability Distributions of Instance Specifics

start their treatment in weeks 0 to 3, respectively. Because we plan a total of 4 weeks, we assign only a limited number of DTs to these patients. The release and due dates of these patients equal Monday to Tuesday of the given starting week w_{FDT}.

Instances with $P \in \{9, 15, 25\}$ follow a similar pattern but use only 3, 5, and 5 categories and planning horizons of 10, 15, and 15 days, respectively. Patients in instances with $P \in \{3, 4, 5\}$ all have equal release dates (day 0) and due dates (day 1) and every patient needs the same number of DTs. The total number of DTs for these instances is listed in the results Table 4.8.

cat.	w_{FDT}	N_p^{DT}	r_p	d_p
1	-	$\sim \mathcal{U}(4,5)$	0	$P(d_p = 0) = 0.8$
2	-	$\sim \mathcal{U}(8,10)$	0	$P(d_p = 1) = 0.2$
3	-	$\sim \mathcal{U}(12,15)$	0	
4	0	$\sim \mathcal{U}(16,20)$	0	1
5	1	$\sim \mathcal{U}(12,15)$	5	6
6	2	$\sim \mathcal{U}(8,10)$	10	11
7	3	$\sim \mathcal{U}(4,5)$	15	16

Table 4.6: DT Specifics of Larger Instances

The calculation of tight lower bounds is hardly possible for the RPSP, so we manipulated the activity durations of the DTs of randomly generated instances such that there exists an (optimal) solution to the problem for which no unavoidable idle time of the beam resource is necessary. These optimal solutions also do not contain time window violations. The optimal objective value then consists of the total durations of the DTs, that is, the minimum time the beam is used in total. For non-manipulated instances (see results in Table 4.10), we use the same measure as a lower bound to the problem.

All algorithms have been implemented in C++. The MIP was solved using Gurobi 7.0.2. The experiments have been carried out on the Vienna Scientific Cluster (VSC3), whose compute nodes are equipped with two Intel Xeon E5-2650v2, 2.6 GHz, 8 core CPUs each.

4.5.3 Preliminary Study II – VND Neighborhood Evaluation

We conducted preliminary experiments in order to assess the impact of all six neighborhoods used within the VND part of the ILS. Therefore we formed multiple subsets of the $k_d = 6$ neighborhoods and performed randomized computational tests on 10 problem instances including 35 to 175 patients. 5 instances are randomly generated, while the remaining 5 are again "manipulated" instances with known lower bound. Seven subsets have been chosen such that each subset still operates on all three parts of the solution representation. These subsets are:

1. All $k_d = 6$ neighborhoods N1, N2, N3, N4, N5, and N6
2. Neighborhoods N1, N2, and N3
3. Neighborhoods N1, and N5
4. Neighborhoods N4, and N5
5. Neighborhoods N5, and N6
6. Neighborhoods N2, N3, N4
7. Neighborhoods N2, N3, N6

We investigated two neighborhood sizes: The first, "small" approach evaluates $\sqrt{N}$ neighbors per iteration, with N denoting the total number of potential activities to be scheduled, which we abbreviate to "SN." The second approach allows us to examine P neighbors of the current solution (denoted "LN"). On all 10 instances and two neighborhood sizes (small, "SN" and large, "LN"), 16 replications of the

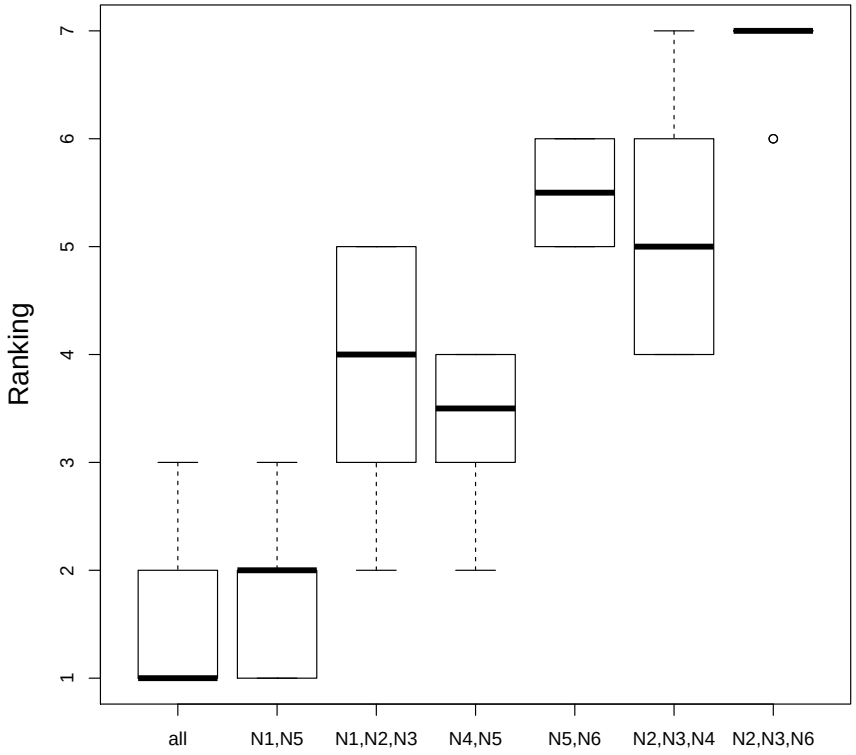


Figure 4.9: Boxplot of seven subsets of neighborhoods containing ranked results averages among 16 replications on 20 different instance-neighborhood size combinations.

ILS were run, equaling 20×16 runs for each of the neighborhood subsets mentioned above. Then, concerning the average results obtained by the 16 replications, the rank of the neighborhood subset was calculated for each of the 10 problem instances and the two neighborhood sizes. The best neighborhood subset obtained rank 1, while the worst one obtained rank 7 (see e.g., Hemmelmayr et al. [2012] for a similar comparison of neighborhood subsets). Figure 4.9 depicts the results of the ranking for all neighborhood subsets in form of box plots. The results show that for some of the tested instances, a subset of the six neighborhoods, more specifically the subset containing N1 and N5, performs best and is therefore ranked number 1. However, averaged over all instances (see the bold line in the boxes representing the median rank), the proposed version of the algorithm containing the full set of neighborhoods, is clearly better than all the remaining subsets.

P	SN/LN	N1	N2	N3	N4	N5	N6
35	SN	44%	31%	18%	17%	49%	32%
35	LN	44%	31%	18%	17%	50%	32%
70	SN	50%	32%	21%	19%	57%	45%
70	LN	52%	32%	21%	20%	59%	43%
105	SN	50%	40%	28%	18%	69%	57%
105	LN	53%	40%	29%	19%	73%	57%
140	SN	52%	40%	35%	19%	77%	64%
140	LN	56%	38%	36%	21%	81%	65%
175	SN	53%	36%	45%	20%	84%	73%
175	LN	58%	34%	48%	23%	86%	74%

Table 4.7: Success rates of each neighborhood during the optimization of five instances and two algorithmic settings (small, large neighborhood) each.

Additionally, we analyzed the success rate of each neighborhood during the optimization by dividing the number of times the neighborhood lead to a better solution through the number of times the neighborhood was called during the optimization, i.e., $n_{success}/n_{called}$. The corresponding results are summarized in Table 4.7. All six neighborhoods have considerable success rates, indicating once more that all six neighborhoods contribute to the search of the best solution. Furthermore, the success rate of each neighborhood apparently increases with the instance size. Hence, larger instances seem to profit even more from the whole range of neighborhoods, which is why we have chosen to include all six neighborhoods in the more extensive experimental tests of small and large instances.

4.5.4 Small Instances: Comparing Heuristics to the Exact Approach

The initial tests entail small instances with known optimal solution (see Table 4.8, column "opt."). We compare results from two versions of the linearized variant of the MIP model presented in Section 4.3: The full model contains all variables and constraints described in Section 4.3, whereas in the reduced model, optional activities are assigned prior to the optimization, thereby radically decreasing the number of variables and constraints in the model (see Table 4.8, columns "vars" and "cons", respectively). The "time-to" column indicates the running time needed to achieve the best found solution (and to prove the optimality of this solution), with a maximum running time for both MIP versions of 24 hours. Furthermore, the two basic versions of the GA and ILS (large neighborhood) are listed. The results show that the optimal solution was found almost for all problem instances in a reasonable time span by the GA and the ILS, with a small advantage for the latter method. The smaller instances with $N^{DT} = \sum_p N_p^{DT} \leq 36$ also could be solved to optimality by Gurobi (marked by ˆ). However, Gurobi was only capable of proving optimality for the two smallest instances (marked with *). Furthermore, the running time to find (the optimal) solutions is already vast for the small instances; those with 45 or more DTs could not be solved to optimality by Gurobi. For instances with 15 and 25 patients, not even a single feasible solution was found after 24 hours of running time. We therefore conclude, that solving the RPSP to optimality using mathematical programming techniques is beyond the scope for larger problem instances.

Small Instances														
			MIP full				MIP reduced				Heuristics		
P	N^{DT}	Opt.	Full	Time-to (s)	Vars	Cons	Reduced	Time-to (s)	Vars	Cons	GA	ILS	Time
3	12	184	*184	4 / 33	580	850	*184	2 / 13	358	508	**184**	**184**	60
4	16	240	*240	82 / 5797	923	1276	*240	1 / 122	623	812	**240**	**240**	60
5	20	248	ˆ248	7609	1226	1647	ˆ248	4369	876	1078	**248**	**248**	60
3	27	414	ˆ414	3326	2596	3818	ˆ414	1184	1359	1411	**414**	**414**	60
5	30	372	ˆ372	22432	3026	4416	376	1518	1745	1704	**372**	**372**	60
4	36	540	ˆ540	79853	4609	6862	ˆ540	54769	4148	6507	**540**	**540**	60
5	45	558	591	3173	6743	9736	563	26121	5567	7836	**558**	**558**	60
9	60	720	759	84633	13250	20364	-	86400	11201	17336	**720**	**720**	1800
15	135	1410	-	86400	57409	72971	-	86400	46785	59587	1414	**1410**	3600
25	225	2295	-	86400	156304	184960	-	86400	120699	140088	2353	**2296**	5400

Table 4.8: Results of Small Instances. * Gurobi has found the optimal solution and optimality was proven. ˆGurobi has found the optimal solution but optimality could *not* be proven by the commercial solver.

4.5.5 Medium and Large Instances: Comparing the Heuristic Approaches

Tables 4.9 and 4.10 summarize the results of computational tests performed using larger problem instances that include 35 to 175 patients. The combination of the day assignment and the in-room set-up and tear-down times complicate the calculation of a strict lower bound, because it makes assessing *unavoidable* idle time on the beam resource virtually impossible. Therefore, we adopt two strategies to evaluate the quality of the solutions. Table 4.9 lists five instances in which we manually manipulated the proportion of time used for set-up, tear-down and treatment so that there exists an optimal solution without idle time on the beam. The calculated gaps for these manipulated instances can be interpreted as optimality gaps. Then, Table 4.10 contains averages over 16 randomly generated instances. Here, we compare the solution to a naive lower bound consisting of the total time the beam is required within the planning horizon, equal to the sum of all irradiation durations of all patients.

The "Initial" column lists the objective value of the best solution among the initial GA population, which also serves as a starting solution to the ILS. The "average" column reveals the average of best found solution fitnesses after the time limit has been reached for all proposed methods. Note, that we limited the number of evaluated neighbors within the local search of the ILS. We investigated two neighborhood sizes: The first, "small" approach evaluates $\sqrt{N}$ neighbors per iteration, with N denoting the total number of potential activities to be scheduled, which we abbreviate to "SN." The second approach allows us to examine P neighbors of the current solution (denoted "LN", abbreviation of large neighborhood).

The results show a slight advantage of the GA and cGAILS for the manipulated instances in Table 4.9. The corresponding gaps from the optimal values increase slightly with the instance size and the underlying problem complexity for the manipulated instances. The randomly generated instances in Table 4.10 clearly show the benefits of combining GA and ILS, which for all problem sizes delivers the best solutions. For the real-world, non-manipulated instances, the gap from the lower bound remains more or less stable, even with increasing problem size.

Large Instances												
				Initial		GA		ILS			cGAILS	
P	Av. N^{DT}	Opt.	Time Limit	Average	Gap	Average	Gap	Av. SN	Av. LN	Gap	Average	Gap
35	400	3920.0	7200	5143.0	31.2%	4153.9	6.0%	**4065.3**	4068.2	3.7%	4097.8	4.5%
70	800	7580.0	14400	10515.8	38.7%	**8143.6**	7.4%	8209.3	8196.4	8.1%	8163.3	7.7%
105	1200	11240.0	21600	14463.8	28.7%	12279.2	9.2%	12507.1	12457.9	10.8%	**12252.0**	9.0%
140	1600	14900.0	28800	20309.6	36.3%	**16426.7**	10.2%	16792.3	16630.8	11.6%	16490.1	10.7%
175	2000	18560.0	36000	27057.9	45.8%	**20631.7**	11.2%	21104.4	20956.6	12.9%	20686.4	11.5%

Table 4.9: Results of Large, Manipulated Instances (one instance per P with 16 replications each). Best found solutions over all solution methods are in bold.

Large Instances												
				Initial		GA		ILS			cGAILS	
P	Av. N^{DT}	LB	Time Limit	Average	Gap	Average	Gap	Av. SN	Av. LN	Gap	Average	Gap
35	400	3717.6	7200	5145.7	38.4%	4544.2	22.2%	4476.2	4473.3	20.3%	**4459.6**	20.0%
70	800	7107.4	14400	10522.0	48.0%	8638.9	21.5%	8577.7	8572.0	20.6%	**8520.6**	19.9%
105	1200	10603.0	21600	14442.9	36.1%	13070.1	23.3%	12995.2	12997.9	22.6%	**12820.4**	20.9%
140	1600	13976.7	28800	20315.3	45.4%	17212.8	23.2%	17267.1	17369.9	23.5%	**16930.7**	21.1%
175	2000	17734.8	36000	27016.5	52.3%	22205.6	25.2%	22288.7	22422.1	25.7%	**21946.5**	23.7%

Table 4.10: Results of Large, Real-World-Inspired Instances (16 instances per P with 16 replications each). Best found solutions over all solution methods are in bold.

Figure 4.10 compares the empirical cumulative distribution functions (ECDF) of the four methods: GA, ILS with small neighborhood, ILS with large neighborhood, and cGAILS for one specific instance including 175 patients. For these functions, all 16 replications per solution method were sorted according to their objective value. The graph displays the percentage of solutions (y-axis) among these 16 replications per method that lie below a given objective value (x-axis), i.e., the midpoint of the y-axis gives the median performance of the algorithms, whereas the upper line gives the worst case and the lower line the best case performances. According to this analysis, cGAILS performs better for all percentiles of the ECDF. Although GA and ILS-small depict comparable median performance, both the best and the worst solutions of the GA are weaker than the best/worst of the ILS with small neighborhood. Furthermore, ILS with large neighborhood does not yield comparable results for this specific instance, which is also evident for instances with 140 to 175 patients in Table 4.10.

Table 4.11 gives an overview of some attributes of the achieved solutions by the three metaheuristic approaches for the 175 patients instances (last row in Table 4.10). The "Av. Fit." column denotes the average fitness of the solutions, with "Av. Pen." indicating the penalty term of the fitnesses. The "Av. Holes" column lists the total number of unassigned days over all patients, and "Av. PS" sums the particle switches that cause sequence-dependent set-up on the beam resource. Furthermore, "Av. SR2" indicates how often the same room is required for a DT consecutively, leading to immediate idle time on the beam during the in-room set-up and tear-down times. Finally, "Av. SR1b" counts cases in which it was not possible to iterate through all rooms, so only a switch between two rooms occurred (e.g., room 1, room 2, and again room 1, with only one patient treated in a different room between the two times in the same room). We aim to minimize key figures Av. PS to Av. SR1b to achieve solutions with less idle time. A low number of unassigned days ("holes" within a patient's treatment plan) does not necessarily lead to better solutions; instead, we anticipate some "optimal" number of unassigned days for each instance.

Comparing the three methods according to the key figures gives an initial impression of the advantages and drawbacks of each method. The average GA solution includes only small time window penalties and particle switches, but the same room is more often used consecutively, leading to higher idle time on the beam. In contrast, ILS results in a higher number of particle switches, simultaneously reducing Av. SR2 to only 27.8. Then cGAILS can further shorten this number to 25.6, at the same time accepting somewhat more particle switches than the GA solutions. However, particle switches cause beam idle time of 3 minutes, whereas using the same room twice consecutively, on average, leads to an idle time of 17.9 minutes. Therefore, if idle time is unavoidable, we would rather switch beam types

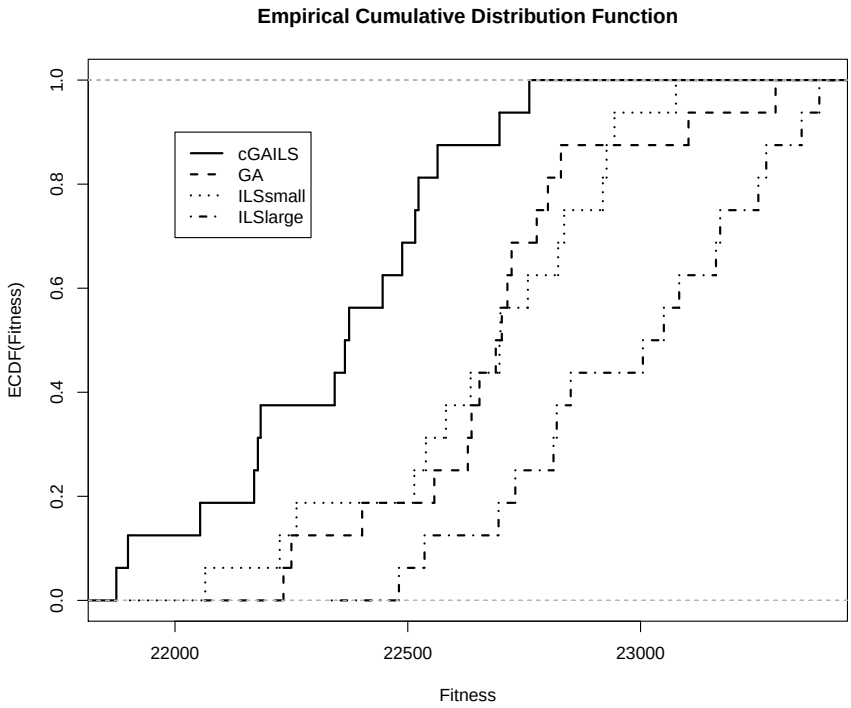


Figure 4.10: Empirical Cumulative Distribution Functions: 175 patients and 16 replications.

Method	Av. Fit.	Av. Pen.	Av. Holes	Av. PS	Av. SR2	Av. SR1b
GA	22205.6	28.1	44.9	134.4	33.4	509.2
ILS-small	22288.7	37.4	48.3	305.5	27.8	564.9
cGAILS	21946.5	26.0	54.3	179.7	25.6	535.2

Table 4.11: Key Figures of 175-Patient Instances

than use the same room twice in a row. Finally, the average number of holes within the best found solution is significantly higher in cGAILS than in GA and ILS. Accordingly, we conclude that the optimal number of unassigned days over all patients lies between 50 and 60 for the 175-patient instances.

4.5.6 Statistical Tests

To prove the superiority of the hybrid method cGAILS relative to the stand-alone methods GA, ILS with small neighborhood, and ILS with larger neighborhood, we conducted various statistical tests, the results of which are presented in the following tables. We performed both t-tests (two-sided) and Wilcoxon Rank tests and obtained comparable results, as both tests support the hypothesis that the hybrid method achieves significantly better outcomes. ***, **, and * denote p-values ≤ 0.001, ≤ 0.01, and ≤ 0.05, respectively, whereas "ns" indicates cases with p-value > 0.05. For each of the $5 \times 16 = 80$ instances, 16 replications were run.

The dominance of the hybrid method over the stand-alone methods, for all $5 \times 16 = 80$ real-world-inspired instances, can hence be verified statistically. Specifically, these tests yield significantly better results for almost all instances: cGAILS produces significantly superior results (p-value ≤ 0.01) to GA in at least 78 of the 80 instances. Furthermore, the combined method performs significantly better than ILS with small neighborhood in 61 and better than ILS with large neighborhood in 62 cases.

t-Test (two-sided)												
	cGAILS vs. GA				cGAILS vs. ILS-SN				cGAILS vs. ILS-LN			
P	***	**	*	ns	***	**	*	ns	***	**	*	ns
35	16	16	16	0	2	5	6	10	1	2	5	11
70	16	16	16	0	4	9	13	3	4	10	12	4
105	16	16	16	0	12	14	15	1	12	12	14	2
140	15	16	16	0	15	16	16	0	16	16	16	0
175	4	11	14	2	9	11	11	5	12	14	15	1
$\sum_P$	67	75	78	2	42	55	61	19	45	54	62	18

Table 4.12: Results of t-Tests of 80 Instances, Batched by Number of Patients P per Instance.

Wilcoxon Rank test												
	cGAILS vs. GA				cGAILS vs. ILS-SN				cGAILS vs. ILS-LN			
P	***	**	*	ns	***	**	*	ns	***	**	*	ns
35	16	16	16	0	2	4	6	10	0	2	4	12
70	15	16	16	0	3	8	12	4	4	8	12	4
105	16	16	16	0	9	13	16	0	12	12	15	1
140	13	16	16	0	14	16	16	0	16	16	16	0
175	4	8	15	1	9	10	11	5	12	14	15	1
$\sum_P$	64	72	79	1	37	51	61	19	44	52	62	18

Table 4.13: Results of Wilcoxon Rank Tests of 80 instances, Batched by Number of Patients P per Instance.

4.6 Summary

Scheduling recurring radiotherapy treatment appointments in ion beam facilities in which multiple rooms share one particle beam represents a complex job shop scheduling problem with custom constraints. We have introduced the specific problem setting and formulated the problem mathematically. However, in realizing that solving the MIP for real-world instances is intractable, we developed a GA and an ILS approach, both of which build on a multi-encoded solution representation and a chronological decoding algorithm. The GA contains tailor-made, feasibility-preserving, crossover operators and an offspring selection strategy. The local search within the ILS is composed of a VND that operates on six different neighborhoods of the incumbent solution.

We have shown that the two stand-alone metaheuristic approaches lead to excellent solutions for small problem instances, even highly dominating the MIP approach with regard to running times and solution quality. For larger instances, the combination of the population-based GA and the individual-based ILS leads to significantly better results than each of the approaches achieves individually, yielding solutions that perform soundly on all measured key figures.

Chapter 5

Stochastic Radiotherapy Scheduling

5.1 Motivation

In general, radiotherapy treatment appointments are planned a few days or weeks in advance and emergency patients who need to receive treatment immediately are rare. Nevertheless, real-world data uncovers high uncertainty in treatment durations for radiotherapy appointments, even though medical physicists are able to accurately estimate the planned irradiation duration during the intense treatment planning process. Uncertainty is a key challenge in any appointment scheduling process (Gupta and Denton [2008]), but the underlying uncertainty in radiotherapy treatment durations has not yet been considered in the radiotherapy appointment scheduling literature (see Section 4.2).

Even though waiting time to the first treatment appointment (i.e., the start of the recurring treatment process) has been addressed as a crucial objective in many academic papers on radiotherapy appointment scheduling, daily waiting time between the planned treatment start and the actual time treatment is performed has not been considered so far. However, a small waiting time is especially important to chronically ill people, who must undergo not just one treatment appointment but several, consecutive appointments. For example, patients receiving radiotherapy experience multiple irradiation treatments on consecutive days. As many patients are treated consecutively in a treatment facility, typically a delay of one single patient affects the starting times of various successive patients and in the worst case causes waiting time for all upcoming patients on a given day. Practitioners, however, tend to focus on the optimization of resource usage solely as was the case in Chapter 4, due to high fixed costs associated with machines and staff, such that they prefer tight schedules and may underestimate actual treatment durations. But

uncertainty in appointment durations might cause resource-use conflicts as well, and this effect tends to increase for tighter schedules (Gupta and Denton [2008]).

Therefore, to enhance patient satisfaction and wellbeing, we strive to find a method that optimizes patient waiting time and resource usage simultaneously, and increases robustness in the baseline schedules while also considering the stochasticity of the appointment durations. We show that for the highly constrained and stochastic problem of radiotherapy appointment scheduling in special ion beam facilities where mostly only one beam resource is available, it is beneficial to insert activity time buffers. Our newly proposed buffer concept is suitable for any distribution function of the treatment durations and features a buffer parameter using percentiles of the distribution to determine the actual planned activity duration. Then, after optimizing this buffer parameter for our specific distribution functions and patient mix, we compare three algorithmic approaches: a deterministic variant, a stochastic variant, and a quasi-deterministic variant. Each is advantageous in different circumstances.

This chapter is organized as follows: Section 5.2 presents the problem statement of the stochastic radiotherapy appointment scheduling problem and discusses the constraints and objectives, as well as the estimation of the underlying probability distributions of the activity durations. Section 5.3 continues with the mathematical modeling formulation of the described short-term problem. Section 5.4 reviews related work on stochastic appointment scheduling, and general strategies to deal with uncertainty. Section 5.5 is dedicated to the methodology used: We describe the buffer concept in detail and explain the genetic algorithm we used to find good baseline solutions, the variants of evaluating the solution, and the reactive procedure that mimics everyday decisions by human decision makers in pre- or postponing activity starting times in response to deviations from the baseline schedule. The results of our intensive computational tests are in Section 5.6. Finally, Section 5.7 concludes and proposes some possible directions for further research.

5.2 Detailed Problem Description

5.2.1 Baseline Constraints

The current research problem is highly related to the one described in Chapter 4, but multiple assumptions differ. First, we consider a short-term problem with a planning horizon of only $D = 5$ days, corresponding to a week from Monday to Friday. Second, we do not consider optional appointments and third, we assume the resources required for the appointments to be known beforehand so no alternative resources are available.

During the five days, patients $p \in \mathcal{P}$ must attend a predefined number of

irradiation treatments. The treatment pattern, determined by the days of treatment and days without treatment, depends on the type of patient. We map those treatment patterns to binary vectors of length 5, where a vector component with value 1 indicates a treatment day and a value of 0 indicates a day without treatment. Accordingly,

1. Patients finishing treatment in the given week have between 2 and 5 irradiation treatments left. They must engage in treatment activity on each day starting on Monday until the number of missing treatments is met such that a patient with 3 missing irradiations follows the pattern $(1, 1, 1, 0, 0)$.

2. Patients starting their treatment in the given week need to schedule between 3 and 5 treatments. Their forced treatment activity starts backward on Friday, until the number of planned treatments is reached. A patient with 3 planned treatments thus starts treatment on Wednesday, followed by treatments on Thursday and Friday, and the corresponding treatment pattern is $(0, 0, 1, 1, 1)$.

3. All other patients have either 4 or 5 treatments scheduled in the current week. If a patient has 4 treatments scheduled, the earliest weekday on which a treatment break is allowed must be established too. Fore example, a patient with an earliest break on Wednesday could follow one of the following patterns: $(1, 1, 0, 1, 1)$, $(1, 1, 1, 0, 1)$, or $(1, 1, 1, 1, 0)$. The earliest break is mostly defined by the previous week's schedule, in that within a span of five consecutive days, a patient needs to receive at least four irradiation treatments.

Therefore, the treatment pattern is predefined for patients starting or finishing treatment, as well as for patients with treatments in the given week and patients for which the earliest possible break day is Friday. For patients with 4 treatments to be scheduled and the earliest possible day off treatment being Thursday or before, the optimization algorithm needs to determine the treatment pattern and which of the possible weekdays should be the one without treatment.

Again, a daily treatment (DT) consists of three inseparable sub-activities, which we here schedule independently: in-room preparation of the patient, irradiation, and post-irradiation exiting. As mentioned before, two patients who need the same treatment room should not be scheduled consecutively, because doing so leaves unavoidable idle time for the particle beam while patient 1 exits and patient 2 gets prepared. Therefore, schedules alternating between treatment rooms are preferable, though not always possible, because patients are not evenly distributed among treatment rooms. The starting time of the N_p^{DT} treatments to be scheduled for patient p also is now constrained by the patient's preferences: Each patient has a predefined time window, defined by his or her individual daily release times and due times, abbreviated by r_p and d_p, respectively, during which the irradiation should

take place. Deviations from this time window, including premature or belated starting times, lead to a penalty in the objective function.

5.2.2 Stochasticity and Estimating the Probability Distributions

The duration of radiotherapy activities is highly stochastic:

- The in-room preparation might take considerably longer than expected if the positioning of the patient fails and a second attempt is necessary. Furthermore, patients are sometimes less mobile and need additional help when entering the room.

- If a patient moves during the irradiation or needs a break, the irradiation needs to be either interrupted or even aborted. An interruption leads to an extension of the planned activity duration; aborting the irradiation leads to a shorter than planned activity.

- Machine error or room unavailability might cause an activity to take longer than expected. For example, a room might need to be cleaned after a patient exit, which delays the start of the next patient's treatment in the same room.

To estimate the underlying probability distributions of the three radiotherapy activities we analyzed real-world data from 113 patients and 2,270 irradiation appointments. The data were collected in a newly opened ion-beam therapy center MedAustron in Wiener Neustadt, Austria, among patients treated in 2017. Actual activity durations were recorded for these patients, so we can fit a distribution function to the data. We use Easyfit 5.6, which allows for automated fit of many distributions to the data and an easy and fast comparison of different models.[1]

We assume that the preparation and exiting activities of all patients follow the same distribution, as in Figure 5.1. The best fitting distribution function for these two types of activities is of the family of Burr distributions, with the probability density function

$$f(x) = a \cdot k \cdot (\frac{x}{b})^{a-1} \cdot \frac{1}{b \cdot (1 + (\frac{x}{b})^a)^{k+1}} \tag{5.2.1}$$

and the parameters in Table 5.1. Here, b is a scale, and a and k are the shape parameters. For the irradiation activities, we clustered the patients into four groups according to treatment complexity and planned activity durations, as in Figure

[1] http://www.mathwave.com/easyfit-distribution-fitting.html

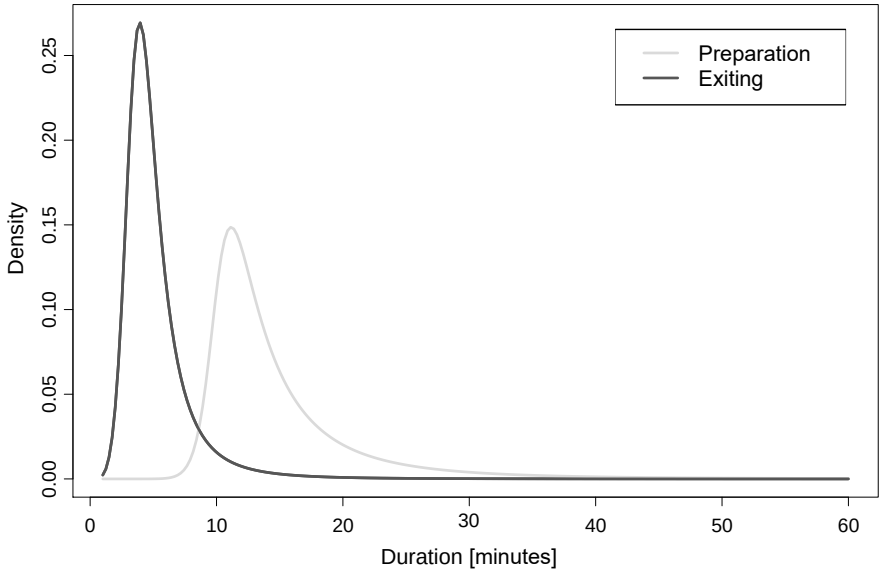


Figure 5.1: Distribution of Duration of Preparation and Exiting Activities.

5.2. In this case, Dagum distributions fit the data best. Again, the distribution parameters are to be found in Table 5.1. The corresponding density function is:

$$f(x) = a \cdot k \cdot (\frac{x}{b})^{a \cdot k - 1} \cdot \frac{1}{b \cdot (1 + (\frac{x}{b})^a)^{k+1}}. \tag{5.2.2}$$

King [2017] provides details on the family of Burr (and Dagum) distribution functions. All distributions are asymmetric and right skewed, reflecting the comparably large probability of outliers that exhibit considerably higher duration than the expected values.

To assess the quality of the distribution fitting, we here depict the empirical cumulative distribution functions (ECDF) in comparison to the fitted, theoretical distribution functions listed in Table 5.1. The poor fit for irradiation group 4 results from the low number of observations in this group (162 irradiation durations) and high number of outliers. Nevertheless we chose to fit a Dagum distribution also for group 4 as we did for groups 1, 2 and 3.

5.2.3 Schedule Execution and Objectives

The constraints listed above need to be respected when constructing a baseline schedule including planned starting times which will be communicated to the patients. However, the actual schedule and corresponding actual appointment starting times are revealed only after the outcome of the random variables. Then, a reactive

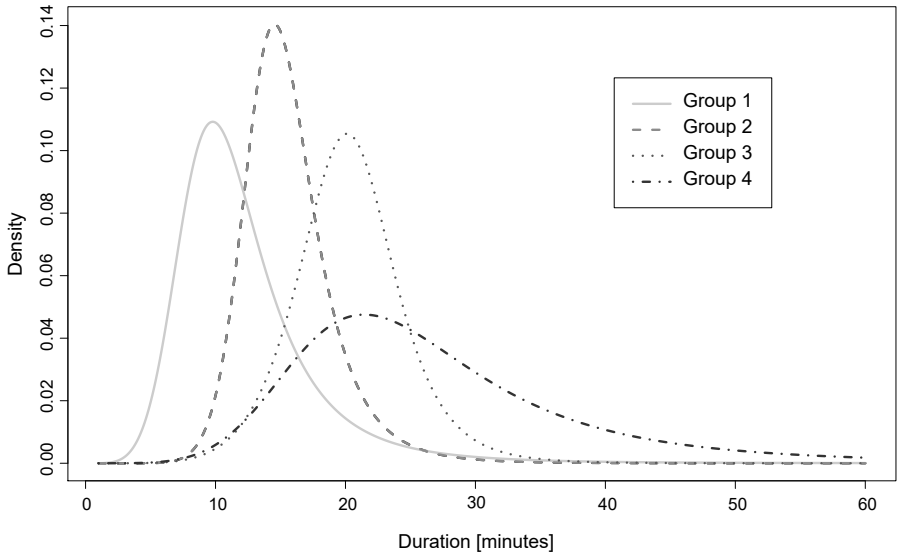
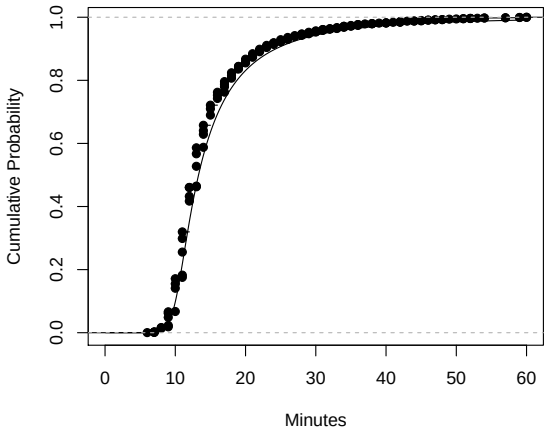


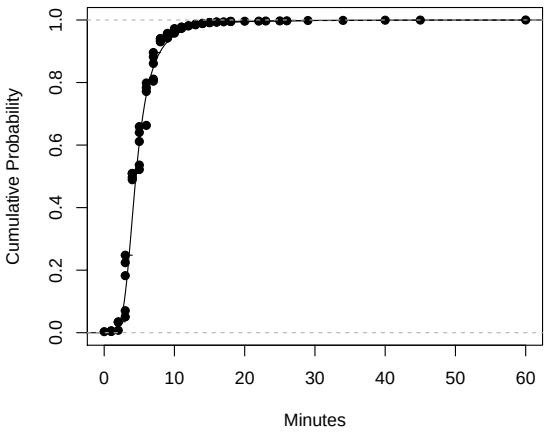
Figure 5.2: Distribution of Duration of Irradiation Activity for 4 Patient Groups.

Act.	Dist	Group	%P	k	a	b	mean	stdev	CV	25%	50%	75%
Pre	Burr	-	100%	0.2	13.4	10.3	15.1	8.9	0.6	11.0	12.9	16.3
Irr	Dagum	1	43%	1.4	4.1	10.0	12.3	5.8	0.5	8.8	11.1	14.3
Irr	Dagum	2	29%	1.3	7.7	14.5	15.6	3.7	0.2	13.3	15.2	17.4
Irr	Dagum	3	22%	0.6	10.1	21.6	20.3	4.4	0.2	17.6	20.2	22.8
Irr	Dagum	4	6%	1.5	3.8	21.4	27.7	14.1	0.5	19.3	24.7	32.3
Ex	Burr	-	100%	0.6	5.3	3.9	5.2	3.2	0.6	3.6	4.5	5.9

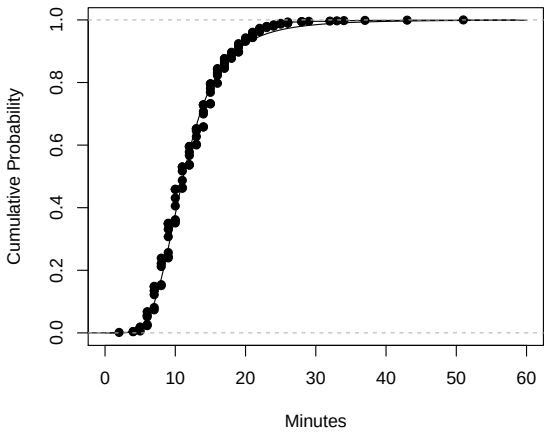
Table 5.1: Properties of fitted distributions for preparation (Pre), irradiation (Irr), and exiting (Ex) activities. "Act." gives the activity type, "Dist" describes the distribution family, "Group" is the patient group and, "%" is the corresponding probability of a patient belonging to the given group according to estimates by the facility. Whereas "k," "a," and "b" are the distribution parameters, "mean," "stdev," and "CV" the mean, standard deviation, and coefficient of variation of the distribution, respectively. The columns "25%," "50%," and "75%" include the quartiles of the distribution.



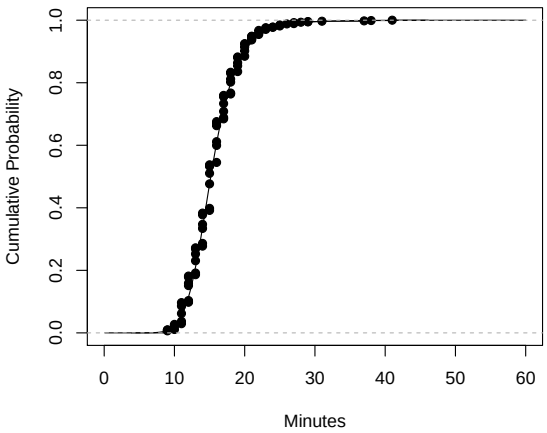
(a) Preparation Activity



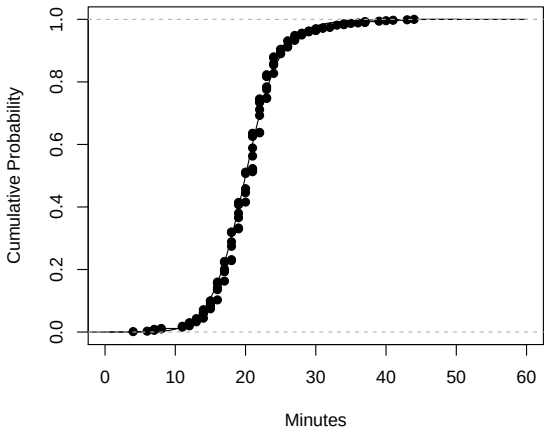
(b) Exiting Activity



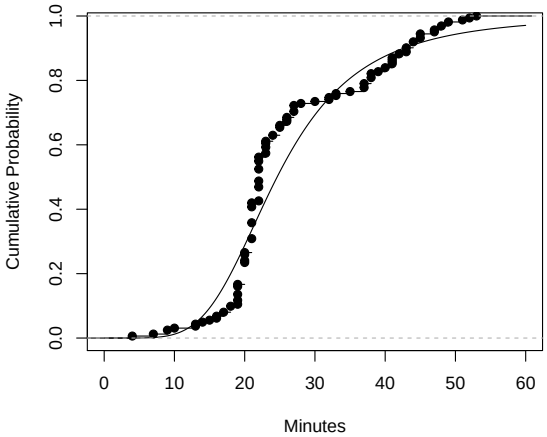
(c) Irradiation Group 1



(d) Irradiation Group 2



(e) Irratiation Group 3



(f) Irradiation Group 4

Figure 5.3: Comparison of Empirical Cumulative Distribution Functions (ECDFs) (dots) and Theoretical Distribution Functions (lines) for the Activities and Fitted Distributions from Table 5.1.

procedure can determine how to respond to deviations from the planned schedule. This reactive procedure mimics the human planner's behavior, as described in detail in Section 5.5.5.

The objective is formed by three highly correlated parts that are weighted by the parameters λ_1, λ_2 and λ_3. All three objective function components are measured in minutes and need to be be minimized, including

1. The *actual* beam active time for each day d, which is defined by the finish time of day d's last activity on the beam resource, denoted by f_d.

2. The time window violations, denoted by γ_{pd}, which are treatment appointments that are scheduled past their latest starting time d_p on day d.

3. The *actual* waiting time of all patients $p \in \mathcal{P}$.

The patient waiting time consists of two main parts: pre-preparation waiting time, reflecting any delay in the start of the preparation activity, abbreviated with δ^h_{pd} and, pre-beam waiting time, or the time span between the completion of the in-room preparation activity and the actual start of the irradiation activity (ω^h_{pd}). The latter is particularly important, because during the in-room preparation, the immobilized and positioned patient is waiting for the irradiation treatment to start in a sometimes uncomfortable position. Minimizing this time span is crucial.

Evaluating the objective analytically is intractable, so we approximate the objective function by the average objective of H scenarios as in (5.2.3), using f^h_d to denote the actual beam finish time in scenario h and F_h abbreviating the objective function of a single random scenario h as in (5.2.4):

$$\text{minimize} \quad \frac{1}{H} \cdot \sum_{h=1}^{H} F_h \tag{5.2.3}$$

$$F_h = \lambda_1 \cdot \sum_{d=1}^{D} f^h_d + \lambda_2 \cdot \sum_{d=1}^{D} \sum_{p=1}^{P} \gamma_{pd} + \lambda_3 \cdot \sum_{d=1}^{D} \sum_{p=1}^{P} (\delta^h_{pd} + \omega^h_{pd}) \tag{5.2.4}$$

The two types of waiting times are visualized in Figure 5.4: Patient P1's irradiation treatment took longer than expected, which delayed P2's irradiation activity. However, when P2 started the in-room preparation, the delay of P1 was not foreseeable, and P2 had to wait for the start of the irradiation activity. In addition, P1's exit was extended. Therefore, P4 could not enter room 1 on time, leading to a delayed start of P4's preparation activity. Note, that P3's preparation started later than expected as well, considering P3 was supposed to start preparation by the time P1 finished the irradiation. The delay of P1's irradiation finish has a direct impact on P3's preparation starting time. Section 5.5.5 details different reactions to deviations from the baseline schedule.

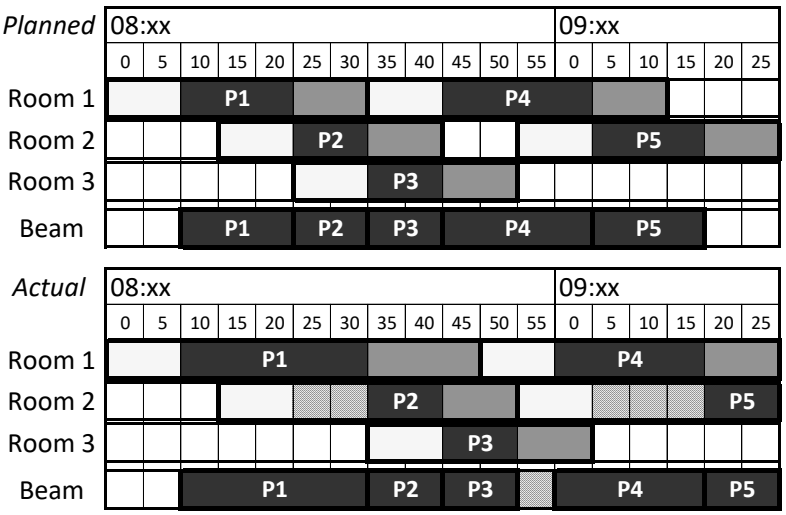


Figure 5.4: Exemplary Planned Schedule and Actually Executed Schedule (white: preparation times; dark gray: irradiation times; medium gray: exiting times); light gray: pre-beam waiting times.

5.3 Mathematical Modeling Formulation

The objective function and constraints described in Section 5.2 can be formulated mathematically. Here, we distinguish between the deterministic variant of the problem, where activity durations are assumed to be known in advance, and the stochastic counterpart. Table 5.2 lists the symbols and sets used in the formulation of the problem, Table 5.3 summarizes all necessary input information and Table 5.4 gives an overview of the decision variables of the mathematical modeling formulation.

Sets	
Notation	Description
$\mathcal{P}$	Set of all patients, $p \in \{1, ..., P\}$.
$\mathcal{D}$	Set of days in the planning horizon, $d \in \{1, ..., D\}$.
$\mathcal{I}$	Set of activities, $i \in \{1, 2, 3\}$.
$\mathcal{R}$	General set of resources, index $r \in \{1, ..., R\}$.
$\mathcal{R}^{pi}$	Set of required resources for activity i and patient p.
$\mathcal{R}^{Room}$	Set of room resources.
$\mathcal{R}^{Beam}$	Set of beam resource.
$\mathcal{H}$	Set of scenarios, $h \in \{1, ..., H\}$.

Table 5.2: Sets of the Mathematical Modeling Formulation

Inputs	
Notation	Description
t_p^i	Planned duration of activity i of patient p including possible buffer.
$\theta_{pd}^{i(h)}$	Actual duration of activity i of patient p on day d (in scenario h).
u_{pqr}	Set-up time between patient p and patient q on resource r.
N_p^{DT}	Number of daily treatment sessions for patient p in the planning horizon.
A_{pd}	Forced treatment day for patient p on day d – binary.
M	Very large number.
$\lambda_1, \lambda_2, \lambda_3$	Objective function weights for beam time, stable time penalties and waiting time, respectively.
r_p	Daily release time of patient p.
d_p	Daily due time for patient p.

Table 5.3: Inputs to the Mathematical Modeling Formulation

Variables	
Notation	Description
s^i_{pdr}	Planned starting time for patient p's activity i on day d at resource r.
$\bar{s}^i_{pd}$	Planned starting time for patient p's activity i on day d.
σ^i_{pdr}	Actual starting time for patient p's activity i on day d at resource r.
$\bar{\sigma}^i_{pd}$	Actual starting time for patient p's activity i on day d.
y_{pqrd}	Binary variable for immediate successor of patient p, namely patient q on machine r on day d.
a_{pd}	Binary variable indicating, whether a daily-treatment activity takes place on day d or not for patient p, i.e., if $\bar{s}^2_{pd} > 0$, then $a_{pd} = 1$, else $a_{pd} = 0$.
f_d	Planned finish time of last activity on the beam resource on d.
$f^{(h)}_d$	Actual finish time of last activity on the beam resource on d (in scenario h).
γ_{pd}	Stable time violation for patient p on day d.
$\delta^{(h)}_{pd}$	Pre-preparation waiting time for patient p on day d (in scenario h).
$\omega^{(h)}_{pd}$	Pre-irradiation waiting time for patient p on day d (in scenario h).
I_h	Random outcomes of the appointment durations for scenario h.
F_h	Objective function given scenario h and the corresponding random numbers I_h.

Table 5.4: Variables of the Mathematical Modeling Formulation

5.3.1 Deterministic Variant

$$\text{minimize} \quad \lambda_1 \cdot \sum_{d=0}^{D} f_d + \lambda_2 \cdot \sum_{d=0}^{D} \sum_{p=0}^{P} \gamma_{pd} \tag{5.3.1}$$

Subject to:

$$f_d \geq \bar{s}^2_{pd} + t^2_p \quad \forall p \in \mathcal{P}, d \in \mathcal{D} \tag{5.3.2}$$

$$a_{pd} \geq A_{pd} \quad \forall p \in \mathcal{P}, d \in \mathcal{D} \tag{5.3.3}$$

$$\sum_{d=0}^{D} a_{pd} \geq N^{DT}_p \quad \forall p \in \mathcal{P} \tag{5.3.4}$$

$$\bar{s}^i_{pd} = s^i_{pdr} \quad \forall i \in \mathcal{I}, p \in \mathcal{P}, d \in \mathcal{D}, r \in \mathcal{R}^{pi} \tag{5.3.5}$$

$$\bar{s}^{i+1}_{pd} = \bar{s}^i_{pd} + t^i_p \quad \forall i \in \{1,2\}, p \in \mathcal{P}, d \in \mathcal{D} \tag{5.3.6}$$

$$\bar{s}^i_{pd} \geq a_{pd} \quad \forall i \in \mathcal{I}, p \in \mathcal{P}, d \in \mathcal{D} \tag{5.3.7}$$

$$\bar{s}^i_{pd} \leq a_{pd} \cdot M \quad \forall i \in \mathcal{I}, p \in \mathcal{P}, d \in \mathcal{D} \tag{5.3.8}$$

$$\gamma_{pd} \geq r_p - \bar{s}^1_{pd} \quad \forall p \in \mathcal{P}, d \in \mathcal{D} \tag{5.3.9}$$

$$\gamma_{pd} \geq \bar{s}^1_{pd} - d_p \quad \forall p \in \mathcal{P}, d \in \mathcal{D} \tag{5.3.10}$$

$$\sum_{q=0}^{P} y_{pqrd} = a_{pd} \quad \forall p \in \mathcal{P}, r \in \mathcal{R}, d \in \mathcal{D} \tag{5.3.11}$$

$$\sum_{p=0}^{P} y_{pqrd} = a_{qd} \quad \forall q \in \mathcal{P}, r \in \mathcal{R}, d \in \mathcal{D} \tag{5.3.12}$$

$$s^1_{qdr} \geq s^3_{pdr} + t^3_p - (1 - y_{pqrd}) \cdot M \quad \forall r \in \mathcal{R}^{Room}, p, q \in \mathcal{P}, d \in \mathcal{D} \tag{5.3.13}$$

$$s^2_{qdr} \geq s^2_{pdr} + t^2_p + u_{pqr} \cdot y_{pqrd} - (1 - y_{pqrd}) \cdot M \quad \forall r \in \mathcal{R}^{Beam}, p, q \in \mathcal{P}, d \in \mathcal{D} \tag{5.3.14}$$

The deterministic objective function (5.3.1) minimizes the operation time and thereby the idle time of the beam while simultaneously minimizing penalties arising from stable time violations for patient p and day d. As in the deterministic variant the extent of waiting time is not known, it is neglected in the objective.

The active time of the beam is calculated according to Constraints (5.3.2). Constraints (5.3.3) assure that a forced treatment for patient p takes place on day d, while Constraints (5.3.4) guarantee that the minimum number of treatments per patient is respected. Constraints (5.3.5) assigns the exact same starting times

$\bar{s}^i_{pd}$ to all resources required for activity i of patient p. All activities 1 to 3 of one patient need to be scheduled tightly together. Therefore, the start of activity $i+1$ must equal the finish of activity i of a given patient p on day d, as depicted in Constraints (5.3.6). Constraints (5.3.7) and (5.3.8) assign a starting time larger than zero for days d on which a treatment for patient p is planned (i.e., $a_{pd} = 1$), and starting time of zero to days d on which there is no treatment scheduled for patient p (i.e., $a_{pd} = 0$).

Inequalities (5.3.9) and (5.3.10) constrain the daily starting time for each patient to his/her individual time window, i.e., the time between his/her release time r_p and due time d_p, considering also possible time window violations, γ_{pd}. Constraints (5.3.11) and (5.3.12) give the immediate successor structure of activities on resource r. Finally, inequalities (5.3.13) and (5.3.14) constrain the starting of successive activities on the room and beam resources, respectively, and confirm that the sequence-dependent set-up times are respected.

5.3.2 Stochastic Variant

The stochastic variant reflects the scenario-based approach discussed in Section 5.2.3:

$$\begin{aligned} \text{minimize} \quad \frac{1}{H}\sum_{h=0}^{H} \cdot \Big[& \lambda_1 \cdot \sum_{d=0}^{D} f_d^h + \lambda_2 \cdot \sum_{d=0}^{D}\sum_{p=0}^{P} \gamma_{pd} \\ & + \lambda_3 \cdot \sum_{d=0}^{D}\sum_{p=0}^{P} (\delta_{pd}^h + \omega_{pd}^h) \Big] \end{aligned} \tag{5.3.15}$$

Subject to:
Constraints (5.3.3) to (5.3.14) are added to the stochastic model as they appear above. Additionally, the following constraints are adjoined:

$$f_d^h \geq \bar{\sigma}_{pd}^{2h} + \theta_p^{2h} \quad \forall p \in \mathcal{P}, d \in \mathcal{D}, h \in \mathcal{H} \tag{5.3.16}$$

$$\delta_{pd}^h = \max(\bar{\sigma}_{pd}^{1h} - \bar{s}_{pd}^1, 0) \quad \forall p \in \mathcal{P}, d \in \mathcal{D}, h \in \mathcal{H} \tag{5.3.17}$$

$$\omega_{pd}^h = \max(\bar{\sigma}_{pd}^{2h} - (\bar{\sigma}_{pd}^{1h} + \theta_{pd}^{1h}), 0) \quad \forall p \in \mathcal{P}, d \in \mathcal{D}, h \in \mathcal{H} \tag{5.3.18}$$

$$\bar{\sigma}_{pd}^{1h} \geq \bar{s}_{pd}^1 - 15 \quad \forall p \in \mathcal{P}, d \in \mathcal{D}, h \in \mathcal{H} \tag{5.3.19}$$

$$\bar{\sigma}_{pd}^{ih} = \sigma_{pdr}^{ih} \quad \forall i \in \mathcal{I}, p \in \mathcal{P}, d \in \mathcal{D}, r \in \mathcal{R}^{pi}, h \in \mathcal{H} \tag{5.3.20}$$

$$\bar{\sigma}_{pd}^{i+1,h} \geq \bar{\sigma}_{pd}^{ih} + \theta_p^{ih} \quad \forall i \in \{1,2\}, p \in \mathcal{P}, d \in \mathcal{D}, h \in \mathcal{H} \tag{5.3.21}$$

$$\bar{\sigma}_{pd}^{ih} \geq a_{pd} \quad \forall i \in \mathcal{I}, p \in \mathcal{P}, d \in \mathcal{D}, h \in \mathcal{H} \tag{5.3.22}$$

$$\bar{\sigma}_{pd}^{ih} \leq a_{pd} \cdot M \quad \forall i \in \mathcal{I}, p \in \mathcal{P}, d \in \mathcal{D}, h \in \mathcal{H} \tag{5.3.23}$$

$$\sigma_{qdr}^{1h} \geq \sigma_{pdr}^{3h} + \theta_p^{3h} - (1 - y_{pqrd}) \cdot M \quad \forall r \in \mathcal{R}^{Room}, p, q \in \mathcal{P}, d \in \mathcal{D}, h \in \mathcal{H} \tag{5.3.24}$$

$$\sigma_{qdr}^{2h} \geq \sigma_{pdr}^{2h} + \theta_p^{2h} + u_{pqr} \cdot y_{pqrd} - (1 - y_{pqrd}) \cdot M \quad \forall r \in \mathcal{R}^{Beam}, p, q \in \mathcal{P}, d \in \mathcal{D}, h \in \mathcal{H} \tag{5.3.25}$$

The stochastic objective function (5.3.15) additionally considers patient waiting time, i.e., pre-preparation waiting time, δ_{pd}^h and pre-beam waiting time, ω_{pd}^h, respectively and optimizes the average over H scenarios. The actual beam active time f_d is now constrained by the actual starting times of all patients and their actual durations, as displayed in Constraints (5.3.16). Constraints (5.3.17) calculate the pre-preparation waiting time as the difference between the actual starting time and the planned starting time of a patient. Consequently, Constraints (5.3.18) portray the calculation of the pre-beam waiting time as the difference between the actual starting time of the irradiation activity and the actual finish time of the preparation activity of patient p. The maximum possible amount of preponement is fixed to 15 minutes, which is reflected in Constraints (5.3.19).

The subsequent Constraints (5.3.20) to (5.3.25) mirror the deterministic constraints, but for the actual starting times $\bar{\sigma}_{pd}^{ih}$ and σ_{pdr}^{ih} instead of the planned starting times $\bar{s}_{pd}^{i}$ and s_{pdr}^{i}.

5.4 Related Work

This section complements Section 4.2. There, we summarized literature on radiotherapy scheduling, which was dominated by deterministic scheduling approaches. Uncertainty in relation to radiotherapy scheduling is addressed by two papers: Sauré et al. [2012] identify effective policies for allocating demand to still unknown patients using a Markov decision process in an effort to minimize the time patients must wait before the treatment starts. Legrain et al. [2015] also address uncertainty related to the arrival of patients to radiotherapy facilities and develop a hybrid online stochastic optimization algorithm to overcome the unknown. However, to the best of our knowledge, all prior studies of radiotherapy appointment consider activity duration as deterministic.

Various articles on more general appointment scheduling problems in health care have successfully integrated uncertainty into their scheduling algorithms. Hence, we here summarize approaches mentioned in classical appointment scheduling literature, followed by a short overview of methodological approaches to address stochasticity in scheduling problems in general.

5.4.1 Stochastic Appointment Scheduling Problems

Stochasticity in appointment scheduling problems can be twofold: First, the patients to be treated might not be known in advance and instead get dynamically revealed during the planning horizon. For example, emergency patients need to be immediately treated, and planned patients do not show up for their appointments. Second, the appointment duration may be subject to uncertainty (Gupta and Denton [2008]). We concentrate on the latter aspect of the stochastic appointment durations. Ahmadi-Javid et al. [2017] review optimization studies that consider random appointment durations (or "service times"). For the special problem of operating room scheduling, Cardoen et al. [2010] consider studies that incorporate uncertain procedure durations. A more recent review by Samudra et al. [2016] includes a section dedicated to uncertainty.

Belien and Demeulemeester [2004] propose models for building robust cyclic surgery schedules when the procedure duration is stochastic. Robustness also plays an important role in the work of Hans et al. [2008], who use advanced optimization techniques combined with historical data on surgery durations to improve capacity utilization. Denton et al. [2007] also conclude that sequencing patients according to the expected variance of their activity duration achieves the best results when the scheduling involves a single server. However, following this strategy might not be beneficial in highly constrained settings; this concern represents the open problem, we address here. Kaandorp and Koole [2007] propose a local search procedure to optimize a weighted average of patient waiting time and doctor idle time. They assume the activity duration to be exponentially distributed. Their objective function looks similar to our setting, yet our problem is much more constrained.

Klassen and Yoogalingam [2009] question the "dome" concept (i.e., short appointment intervals in the beginning, gradually increasing to the middle of the day, then gradually decreasing). Using simulation optimization, they show that stochastic appointment durations are sometimes handled more robustly using a plateau-dome that extends the middle section. Koeleman and Koole [2012] show that different service time distributions lead to diverse optimal baseline schedules when they use a local search algorithm and consider emergency arrivals. Begen et al. [2012] propose a sampling based approach to address the problem of discrete random appointment durations, which produces a near-optimal solution with high probability in polynomial time. Erdogan and Denton [2013] present two stochastic models considering uncertain durations, as well as no-shows. Tancrez et al. [2013] consider stochasticity in operating rooms in a more strategic decision-making level and present a Markov process which allows to evaluate the impact of stochasticity on various performance measures. Kemper et al. [2014]'s approach to schedule patients for any convex loss function and any service time distribution indicates that customers should be scheduled in non-decreasing order of their scale parameter.

Finally, Berg et al. [2014] propose three solution methods to address the two-stage stochastic problem of scheduling patient appointments on a single stochastic server, extending work by Denton et al. [2007].

5.4.2 Methodological Approaches to Address Stochasticity in Scheduling Problems

Many approaches deal with uncertainty in project and appointment scheduling, as summarized by Herroelen and Leus [2005] in their overview of literature pertaining to reactive, stochastic, fuzzy, and proactive scheduling approaches. Van De Vonder et al. [2005] thoroughly analyze the impact of buffers in project management, and this research group also has considered various combinations of proactive and reactive approaches (e.g., Van De Vonder et al. [2006, 2007], Demeulemeester et al. [2008], Davari and Demeulemeester [2017]). We adopt a proactive approach too, relying on activity buffers. The reactive policy is then to mimic the decision maker and pre- or postpone the activity starting time according to deviations known at the time and predicted impacts of those deviations in the future.

The classical approach to stochastic combinatorial optimization problems (SCOP) is based on (two-stage) mathematical programming or dynamic programming. However, for most real-world problems, these methods are infeasible due to their size and the limited running time. Metaheuristics then offer good alternatives for solving problems marked by uncertainty. A survey of metaheuristics for solving SCOPs (Bianchi et al. [2009]), lists three possible ways to compute objective functions for SCOPs: (1) if closed-form expressions for expected values are available, compute the objective function exactly; (2) if closed-form expressions are not available or their repeated evaluation is too time consuming, use ad hoc, fast approximations; and (3) if the problem is too complex in terms of dependencies, estimate the objective function by simulation. We consider the latter two variants, because we address a highly constrained problem with extensive probabilistic dependencies. The last variant also is known as "sample average approximation" and was applied successfully to SCOPs by Kleywegt et al. [2002] and Mancilla and Storer [2012]. Finally, Juan et al. [2015] review simheuristics and identify multiple subcategories of this term, depending on how much time is spent on simulation and optimization. Their general scheme for solving SCOPs includes a fast simulation process and few replications to estimate the quality of the solution during the optimization (Bianchi et al. [2009]).

The present chapter deals with stochastic appointment durations in the setting of radiotherapy appointment scheduling with the goal to enhance patient satisfaction and increase schedule robustness. We combine various approaches known from stochastic (appointment) scheduling herein. Specifically, we introduce activity

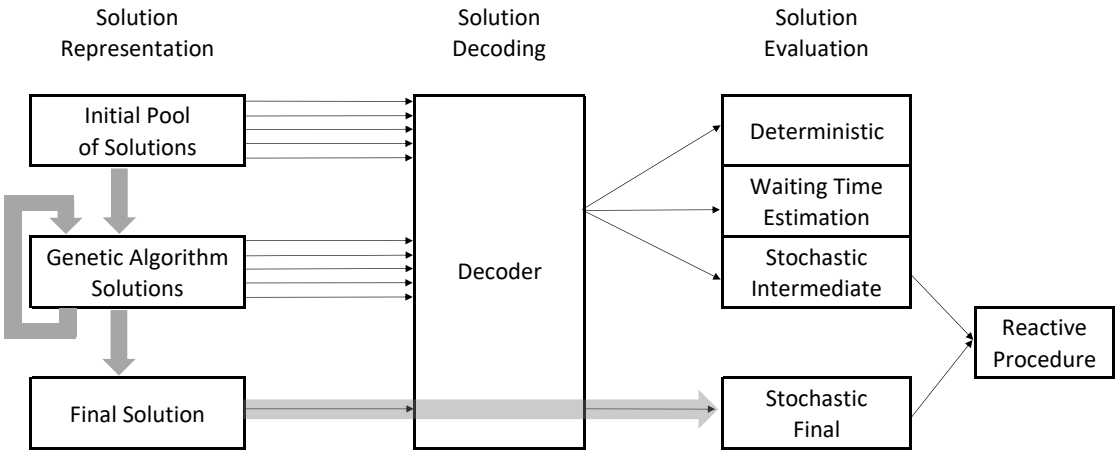


Figure 5.5: Overview of the Solution Method.

buffers to create robust baseline schedules proactively. When evaluating the quality of a solution, we apply a reactive procedure to a sample of scenarios or else use a fast approximation approach.

5.5 Solution Methodology

In this section, we first introduce and define a time buffer to create more robust baseline schedules in Section 5.5.1. Initially, we assume this buffer to be given (see Section 5.6.2 for details on buffer determination). Then we define the formal representation of a solution in Section 5.5.2 and we also discuss how to create an initial solution by applying a simple greedy algorithm. Section 5.5.3 details how we get from the solution representation to a baseline schedule, including planned starting times. In Section 5.5.4, we outline different approaches to evaluating a single solution, followed by a detailed description of the reactive procedure in Section 5.5.5. Finally, in Section 5.5.6 we describe the optimization algorithm we used to find reasonably good baseline schedules.

The methodology described in this paper consists of multiple components, which are depicted in Figure 5.5. First, a pool of initial solutions is generated. Then, we apply a genetic algorithm (GA) metaheuristic to enhance the search for good quality solutions. A solution here is represented by the solution encoding displayed in Figure 5.7. To evaluate the quality of a solution, we first decode the solution representation to a schedule including planned appointment starting times. Then, one out of three possible evaluation methods is applied, some of which require a reactive procedure to estimate the expected objective function. The final solution, i.e., the best among all GA solutions after the predefined time limit is reached, is

then evaluated more intensively within the final evaluation procedure, which again uses the same reactive procedure.

5.5.1 Buffer Concept

We strive to create more robust baseline schedules that offer some degree of protection against disruptions during schedule execution. Therefore, we introduce time buffers (Van De Vonder et al. [2005]) to the planned activity durations. The planned activity duration is equivalent to the 50th percentile or median, of the underlying distribution functions (see Table 5.1) for the different activity types. The activity duration including the time buffer then reflects a global buffer parameter $0.5 \leq \beta < 1.0$, which depends on the corresponding percentile of the distribution. For example, if $\beta = 0.75$, using the Burr-distributed preparation time with parameters from Table 5.1, we create a baseline schedule that features an in-room preparation duration of 16.3 minutes (75% of the Burr$(0.2, 13.4, 10.3)$ distribution) instead of the original 12.9 minutes, resulting in a buffer of 3.4 minutes. In this baseline schedule, the activity durations are longer than expected and can be "eaten up" by activities that take more time than the initially planned 50th percentile of the distribution function.

We chose the distribution percentile as the parameter to determine the planned activity duration, because this value is independent of the distribution. The planned activity duration of activity i for patient p, t_p^i, given the buffer parameter β (i.e., the percentile) can be written as $P(\theta_p^i \leq t_p^i) = \beta$, where θ_p^i is the actual (random) activity duration. The probability that the actual activity duration exceeds the planned duration is $1 - \beta$. Section 5.6.2 details how we find the optimal parameter β for a given objective function.

Figure 5.6 displays the planned activity durations of various buffer parameters β for the different activity types and patient groups. Equal steps of the buffer parameter do not correspond to steady buffer increases whereas for constant increments of the buffer parameter, the resulting increments of the buffer (in minutes) grow, as clearly displayed by the irradiation activity of group 4: An increase from $\beta = 50\%$ to $\beta = 60\%$ raises the planned activity duration by 2.5 minutes (from 24.7 to 27.2 minutes). Increasing β from 60% to 70% leads to an increase in the planned activity duration of 3.1 minutes (from 27.2 to 30.3 minutes), and an increase from 70% to 80% in the buffer parameter prompts the planned activity duration to grow by 4.4 minutes (from 30.3 to 34.7 minutes). Such a trend is typical for percentiles of unimodal distribution functions, such as those displayed in Figures 5.1 and 5.2: If the distribution is unimodal, the probability density function is decreasing for values larger than the mode. Accordingly, the distribution function becomes concave as soon as the mode value is exceeded, which occurs in this case, because the buffer size results from the buffer parameter, through the

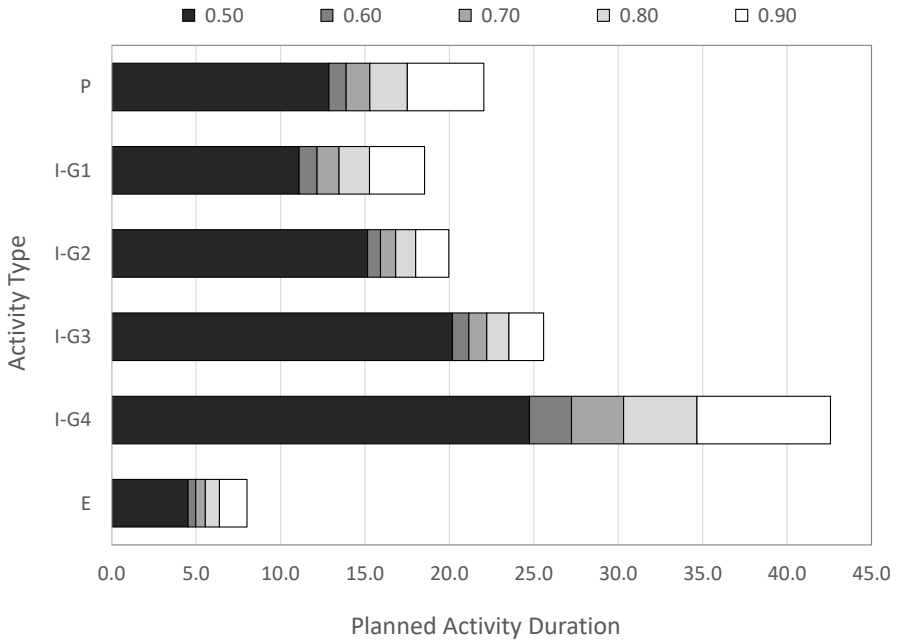


Figure 5.6: Percentiles and Corresponding Planned Activity Durations.

application of the inverse of the distribution function.

5.5.2 Solution Representation and Initial Solutions

Similar to Chapter 4, we establish a multi-encoded solution representation with two major parts: (1) the treatment pattern on a planning horizon of five days for each patient and (2) a vector that displays the sequence by which patients should be decoded on a specific day. The former is the treatment assignment; the latter is the patient sequence, as in Figure 5.7.

For the genetic algorithm we use later in this chapter (see Section 5.5.6), we need

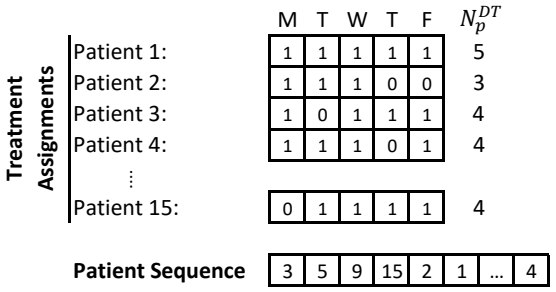


Figure 5.7: Solution Encoding, 15 Patients.

to create several initial baseline schedules (genetic algorithm initial population), generated in a greedy fashion. For each patient we choose a random treatment assignment vector from among the possible treatment patterns. The flexibility in treatment patterns is limited for most patients. Only patients with four treatments in the current week and the earliest break possibility on Thursday or before have multiple patterns from which to choose.

The patient sequence then can be selected in four different heuristic ways to generate a diverse initial genetic algorithm population:

- Sort patients according to their daily due date and try to minimize penalties arising from time window violations. This approach leads to a single patient sequence.
- Ignore time windows and create a pure random starting sequence (9% of initial solutions).
- Ignore time windows and create a starting sequence that avoids treating successive patients in the list in the same treatment room, which causes idle time on the beam resource (10% of initial solutions).
- Sort patients according to their daily due date, which produces a sequence that we call *lst* (latest starting time sequence). Choose a random patient from the top 10% of *lst*, add this patient to the actual patient sequence, and delete the entry from *lst*. Continue this process until the patient sequence is complete and *lst* is empty (80% of initial solutions).

Diversification in the initial "population" of the algorithm enhances the search and is therefore beneficial for the chosen algorithmic setting.

5.5.3 Solution Decoding

Each solution generated in the genetic algorithm is decoded to a baseline schedule, including the planned activity starting times, using a modified version of the algorithm presented in Section 4.4.4 as the current decoding algorithm does not consider optional activities and alternative resource settings. For each day, we generate a prioritized patient list from the global patient sequence that reflects the order in which we chronologically insert patients into the daily baseline schedule. The sequence of patients in this daily list is the same as in the global patient sequence from the solution representation, except that any patients who do not have a planned treatment on that specific day get removed from the daily list. The planned starting times of the activities, $\bar{s}^i_{pd}$, are determined according to Algorithm 7. The three appointment phases (in-room preparation, irradiation, and exiting)

Algorithm 7: Solution Decoding Algorithm.

```
repeat
    Determine next patient p to be scheduled from the patient list of day d;
    s̄¹_pd ← r_p;
    feasible ← false;
    while feasible == false do
        if s̄¹_pd is a feasible starting time for all phases and all resources then
            feasible ← true
        end
        if feasible == false then
            s̄¹_pd ← s̄¹_pd + 1
        end
    end
    Schedule patient p at the determined starting time s̄¹_pd;
    if s̄¹_pd > d_p then
        γ_pd ← s̄¹_pd − d_p
    end
until all activities scheduled;
```

need to be scheduled without idle time when constructing the baseline schedule. That is, we can fix the starting time of the preparation activity and deduce the other starting times from that value. The earliest the activity can start is at the beginning of the patient-specific time window $[r_p, d_p]$, which depends on the release time r_p on each day d (i.e., no scheduling earlier than r_p is allowed). Next, we examine if r_p is a feasible starting time across all required resources over all appointment phases. If so, we fix the starting time and block the resources accordingly. If not, we increment the planned starting time until we find a feasible insertion position. If the final starting time for patient p is greater than the corresponding due time d_p, we face a penalty γ_{pd} in the objective function.

This approach differs from pure chronological scheduling in that we can fill "holes" in the schedule. Holes might occur if a patient with a later release time appears early in the patient list or two patients are assigned successively to the same treatment room, creating idle time for the beam resource. In this second scenario, we might schedule another patient who requires a different treatment room in the interim and thereby minimize beam idle time.

Algorithm 8: StochasticEvaluation(S, H)

input : a baseline schedule S and # of scenarios H
output : estimate for the expected objective function value

```
for h ← 1 to H do
    draw random scenario I_h;
    apply reactive procedure to S using I_h;
    calculate objective value F_h;
end
return 1/H · Σ_{h=1}^{H} F_h
```

5.5.4 Solution Evaluation

We distinguish two solution evaluations: The intermediate evaluation established during the execution of the optimization algorithm, occurs multiple times on several baseline solutions. The final evaluation instead assesses a single, final baseline schedule at the very end of the optimization procedure. The former needs to be fast, whereas the latter may be computationally more expensive. The need for two evaluations comes from the stochastic nature of our model, in which the objective function is a mathematical expectation of a random variable resulting from the actual realization of all random durations. We cannot evaluate this expectation analytically but must resort to a sample average to approximate it. Depending on the sample size, the precision of this estimate can vary. The corresponding mathematical formulation is in Equation (5.2.4). A pseudo code of the stochastic evaluation scheme (STO) is in Algorithm 8.

The final evaluation procedure assesses the quality of the solution by drawing $H = 1,000,000$ random scenarios from the distribution functions listed in Table 5.1. To derive an actual schedule from an encoded solution plus a current random scenario, we introduce a reactive procedure that we apply if the actual activity durations deviate from the planned ones (see Section 5.5.5).

For the intermediate evaluation, we adopt a similar strategy but with fewer random scenarios per solution evaluation. Preliminary tests show, that if the running time is short (e.g., 1 hour), drawing $H = 100$ random scenarios yields a good compromise between evaluation accuracy and running time limits. For longer CPU times, we evaluate $H = 1,000$ random scenarios for each solution. As this stochastic evaluation strategy is computationally expensive, we compare this strategy with two other approaches:

- A deterministic approach (DET) to the problem approximates actual beam active time by the deterministic beam active time of the baseline schedule; it disregards the actual waiting time of the patients. With the optimization

algorithm, we then evaluate each solution:

$$\text{minimize} \quad \lambda_1 \cdot \sum_{d=0}^{D} f_d + \lambda_2 \cdot \sum_{d=0}^{D} \sum_{p=0}^{P} \gamma_{pd} \quad , \tag{5.5.1}$$

where f_d is the beam active time of the baseline schedule.

- A quasi-deterministic variant approximates actual beam active time by the deterministic beam active time of the baseline schedule f_d. To estimate actual waiting time, we leverage the observed correlation between idle time on the beam resource (excluding set-up time to particle type switches) and actual patient waiting time. Using this correlation, we can estimate patient waiting time by fitting a linear regression line of the form,

$$\text{waiting time} = A + B \times \text{schedule idle time}. \tag{5.5.2}$$

The data for this regression are gathered during the optimization by drawing random scenarios for intermediate schedules created by the optimization algorithm given a fixed buffer parameter β, then evaluating those scenarios in detail. The regression gets updated regularly during the execution of the algorithm. This approach constitutes a waiting time estimation strategy (WTE). Figure 5.8 depicts the relationship between the actual patient waiting time and the beam idle time for a random instance with 100 patients and a buffer parameter $\beta = 0.8$. The corresponding regression function to estimate the actual patient waiting time is waiting time $= 2110.1 - 0.3082 \times$ schedule idle time.

5.5.5 Reactive Procedure

The reactive procedure is called for every randomly drawn scenario which needs to be evaluated in detail during the stochastic intermediate or final evaluation. It mimics the human decision maker by pre- or postponing activity starting times when deviations from the baseline schedule occur. The general patient sequence on all resources is retained; only starting times are shifted. All resource related constraints from the baseline schedule need to be met also in the reactive procedure (e.g., each resource still has a maximum capacity of one).

The actual starting time of an activity i of patient p depends on two main factors: the type of activity i, whether preparation, irradiation or exiting, and the starting times of the activities prior to the current activity on the same resource set $\mathcal{R}^{pi} \subset \mathcal{R}$, where $\mathcal{R}$ is the total set of resources, and $\mathcal{R}^{pi}$ is the set of resources required for activity i of patient p. We denote the actual starting time of activity i for patient p on day d as $\bar{\sigma}^i_{pd}$. The reactive procedure then is based on three principles:

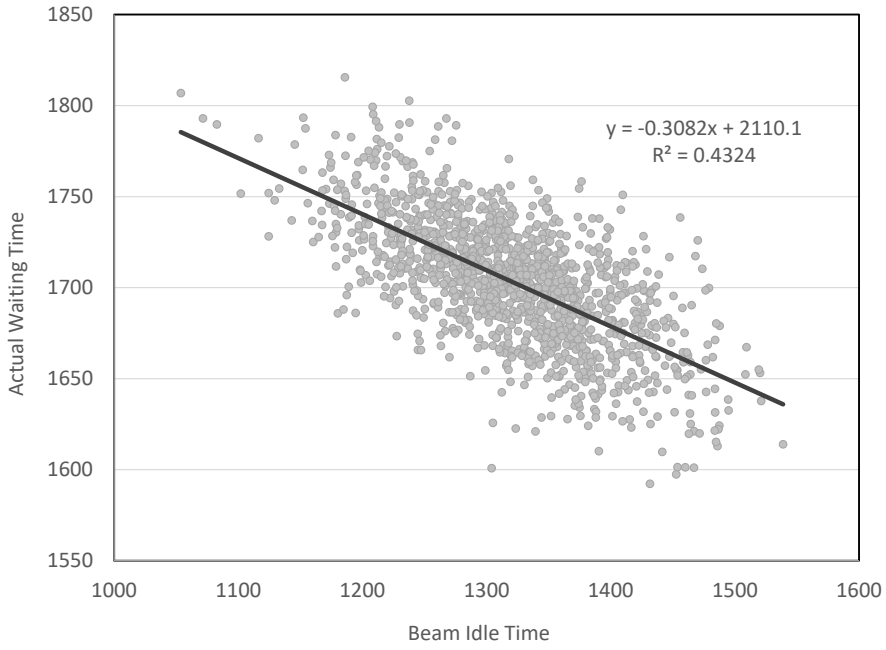


Figure 5.8: Estimation of Actual Waiting Time by Beam Idle Time, Regression Results.

- If the current activity i of patient p on day d is exiting ($i = 3$) and patient p's irradiation activity ($i = 2$) has already finished, start the exiting activity directly after the irradiation activity has finished. The actual starting time then is the sum of the actual irradiation starting time and the actual irradiation duration: $\bar{\sigma}^3_{pd} = \bar{\sigma}^2_{pd} + \theta^2_{pd}$.
- If the current activity i of patient p on day d is irradiation ($i = 2$), and patient p's in-room preparation activity ($i = 1$) has already finished, start the irradiation activity as soon as the previously irradiated patient q has finished the irradiation plus some additional set-up time in case the beam type needs to be switched from patient q to patient p. The actual starting time of the irradiation of patient p is the maximum of the preparation finish time (sum of the actual preparation starting time and actual preparation duration) and the irradiation finish time of the previous patient q (sum of q's actual irradiation starting time plus q's actual irradiation duration and some potential set-up time u_{qpr}): $\bar{\sigma}^2_{pd} = \max(\bar{\sigma}^2_{qd} + \theta^2_{qd} + u_{qpr}, \bar{\sigma}^1_{pd} + \theta^1_{pd})$. If $\bar{\sigma}^2_{qd} + \theta^2_{qd} + u_{qpr} > \bar{\sigma}^1_{pd} + \theta^1_{pd}$, then p's preparation has finished earlier than the beam is available for treatment, and patient p thus faces pre-beam waiting time. Otherwise, we confront beam idle time.
- If the current activity i of patient p on day d is in-room preparation ($i = 1$),

then consider multiple possible cases:

1. If p's preparation activity can only start later than planned, because the room resource is not available at the planned starting time, the preparation shall start as soon as possible.
2. If p's preparation activity might start earlier, we distinguish three scenarios, in which patient q is the patient using the beam resource previous to patient p (but not necessarily in the same room).
   (a) If patient q's irradiation activity started earlier than planned ($\bar{\sigma}^2_{qd} < \bar{s}^2_{qd}$) *and* patient p's planned preparation starting time is greater than q's actual irradiation starting time ($\bar{s}^1_{pd} > \bar{\sigma}^2_{qd}$), it makes sense to prepone p's preparation. The actual starting time is then the maximum among three choices: (i) the earliest room availability; (ii) the patient availability, where we assume a maximum negative deviation from the planned starting time of 15 minutes, before which we assume the patient is not present yet at the facility; and (iii) the planned preparation start minus the earliness of q's beam activity, $\bar{\sigma}^1_{pd} \geq \bar{s}^1_{pd} - (\bar{s}^2_{qd} - \bar{\sigma}^2_{qd})$.
   (b) If scenario (a) does not apply because patient p's preparation starts earlier than patient q's irradiation activity but patient q started preparation earlier than planned (i.e., $\bar{\sigma}^1_{qd} < \bar{s}^1_{qd}$), it still makes sense to prepone p's preparation, even if the estimator of optimal earliness is more insecure in this case. The actual starting time of p's preparation activity is the maximum among the three options: (i) the earliest room availability; (ii) the patient availability ($\bar{\sigma}^1_{pd} \geq \bar{s}^1_{pd} - 15$); and (iii) the planned preparation start minus the earliness of q's preparation, $\bar{\sigma}^1_{pd} \geq \bar{s}^1_{pd} - (\bar{s}^1_{qd} - \bar{\sigma}^1_{qd})$.
   (c) Otherwise, start the preparation of patient p at the planned starting time, $\bar{s}^1_{pd}$.

Figure 5.9 depicts the decision process of the reactive procedure using an example that reflects Case 2. Patient P1's preparation takes less time than scheduled, so P1 starts irradiation as early as possible. For patient P2, Case 2(a) applies: We prepone P2's preparation activity by the same amount that P1's irradiation is preponed (here, 5 minutes). The same strategy applies to P3. The decision about the potential preponement of P4 is more difficult though, because by the time P4's preparation could start, P3's preparation is still ongoing. However, P3 started preparation 5 minutes early, so we suggest preponing P4's preparation by the same amount (Case 2(b)). Finally, P2's exiting activity took longer than expected. Therefore P5's preparation is delayed by 5 minutes (Case 1).

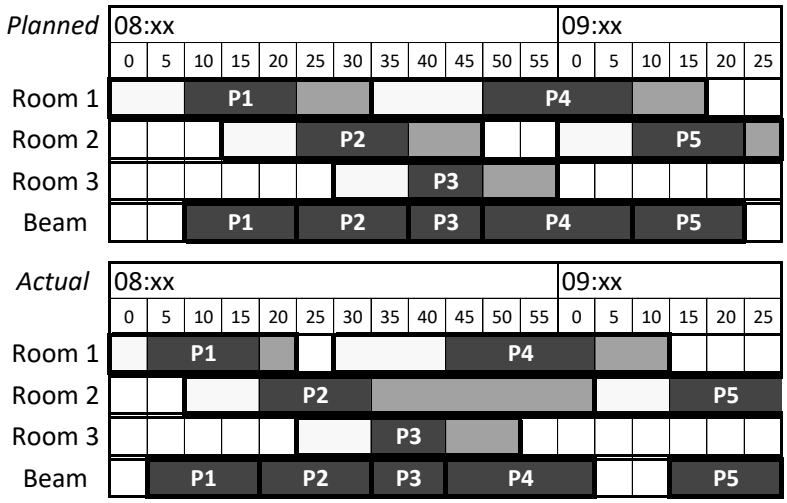


Figure 5.9: Visualization of Reactive Procedure. Top: Baseline schedule including planned activity starting times. Bottom: Actual executed schedule with activity durations and starting times. White: preparation times; dark gray: irradiation times; medium gray: exiting times.

In practice, preponing patients by more than 15 minutes is rarely possible, because they are unlikely to have arrived at the facility so early. Therefore, the sequence of patients cannot be changed without causing some waiting time or stress.

5.5.6 Optimization Algorithm

We once more apply a GA metaheuristic. The main difference to the algorithm applied in Chapter 4 is the different solution representation which contains only 5 days in the planning horizon and does not consider optional activities (see Section 5.5.2). We again use the GA variant published by Affenzeller and Wagner [2004], which includes an offspring selection process that favors individuals that outperform the fitness of at least one of their parents. The main components of the algorithm are as follows (see Algorithm 2 on page 11 for details):

- CreateInitialPopulation (Section 5.5.2): $\mathcal{P}_0$ describes the initial population, and $\mathcal{P}_i$ denotes the population during iteration i. In general, our population size is 100.

- GetElites: A fixed percentage (here 1%) of the previous population's best individuals is directly transferred to the next population.

- C^B: Set of children that do not meet the success criterion. The success criterion in our case is that the fitness of a child needs to be at least as good as the fitness of the worse parent.

- $\{|\mathcal{P}_{i+1}| < 0.7 \cdot |\mathcal{P}_i| \wedge |\mathcal{P}_{i+1}| + |C^B| < 5 \cdot |\mathcal{P}_i|\}$: We aim to build 70% of the new population from children who meet the success criterion. However, the number of reproductive steps is limited to five times the population size. If we fail in building a new population within this limit, we fill up the population with "bad children" from set C^B using function ChooseRandomElement(C^B).

- PerformSelection($\mathcal{P}_i$): With the rank selection operator, we rank individuals according to their fitness and randomly choose an element, considering the rank-weighted probabilities.

- Crossover(p_1,p_2): We create new individuals by crossing over two parent chromosomes. For the patient sequence, we use the well-known position-based crossover. The treatment assignments are randomly inherited by either parent #1 or parent #2.

- Mutate(c): To create more diverse descendants, we apply a mutation operator to 10% of the offspring. A mutation affects both parts of the chromosome; a randomly chosen patient shifts to a random new position in the patient sequence, and the treatment assignment of a single random patient is reset and newly generated.

5.6 Computational Results

This section is dedicated to the results of extensive computational tests on randomly generated problem instances of varying sizes. We begin with a brief description of the instances, optimization parameters, and environment used for the computational study in Section 5.6.1. Then, Section 5.6.2 addresses the problem of determining the optimal buffer parameter β as a strategic decision, and Section 5.6.3 thoroughly compares the solution evaluation procedures from Section 5.5.4 on the different buffer parameters β and objective function weights.

5.6.1 Experimental Set-up

To generate instances for the computational tests, we use data from a newly opened ion-beam therapy center, MedAustron in Wiener Neustadt, Austria. As we noted previously, the data set contains appointment durations for 2,270 appointments collected from 113 patients. Patients in our randomized instances were randomly

Class	N_p^{DT}	Pattern	Instances 1–8	Instances 9–16
1	5	$(1,1,1,1,1)$	44%	20%
2	3	$(0,0,1,1,1)$	4%	7%
3	2	$(1,1,0,0,0)$	6%	6%
4	3	$(1,1,1,0,0)$	6%	6%
5	4	$(1,1,1,1,0)$	4%	8%
6	4	earliest break Thursday	6%	10%
7	4	earliest break Wednesday	8%	12%
8	4	earliest break Tuesday	10%	14%
9	4	earliest break Monday	12%	16%

Table 5.5: Classes of Patients and Corresponding Treatment Patterns.

assigned to groups according to the following rule: 43% of patients belong to group 1, 29% to group 2, 22% are part of group 3, and the rest (6%) constitute group 4. We consider a sequence-dependent set-up time of 3 minutes if two patients with different beam types (protons or carbon ions) are scheduled sequentially on the beam resource. We randomly assign the beam type to patients, such that 50% receive proton therapy and the other 50% are irradiated with carbon ions. The distribution of patients among the three treatment rooms is assumed to be balanced, with a probability of 33% for each room.

For number of treatments and corresponding treatment pattern, we distinguish nine patient classes. The treatment pattern of the first five classes is predefined and can be displayed using a pattern vector with five entries from Monday to Friday, where 1 indicates a treatment day and 0 indicates a day without treatment. The remaining classes all have four treatments to be scheduled, and the patterns depend on the earliest possible break day. The probability of a patient belonging to the classes is in Table 5.5. We generated 16 random instances, such that those with index 1–8 use the first-class probabilities, whereas instances 9–16 draw random classes from taking the second probabilities.

The instance size varies from 30 to 100 patients. The release times for the daily treatments and the time window length are chosen randomly, though the average length of the time window slightly increases with the release time, allowing for more flexibility at the end of each day. Thus the average time window length for the first half of the day is 276 minutes, but it increases to 360 minutes for the second half of the daily planning horizon. Furthermore, we assume that 20% of patients do not have time window preferences and therefore have maximum flexibility for scheduling their treatment start. We test multiple combinations of the objective function

weights $\lambda_1, \lambda_2, \lambda_3$ for the actual beam active time, the time window violations, and the actual patient waiting time, respectively, including the balanced case $\lambda_1 = \lambda_2 = \lambda_3 = 1.0$ and two cases favoring utilized beam capacity over the patient-centered waiting time and time window violations, namely, $\lambda_1 = 1.0, \lambda_2 = \lambda_3 = 0.5$ and $\lambda_1 = 1.0, \lambda_2 = \lambda_3 = 0.1$.

For the GA from Section 5.5.6, we use the following parameters which have proven to be beneficial in preliminary tests as well as in the deterministic setting of the optimization (see Chapter 4): The population size is 100, the mutation rate is 10%, and the elite segment is 1% of the population. All algorithms were implemented in C++, and the experiments were carried out on the Vienna Scientific Cluster (VSC3) equipped with compute nodes with two Intel Xeon E5-2650v2, 2.6 GHz, and 8 core CPUs each. The running time for tests in phase 1 was $P/10$ hours, where P indicates the number of patients in that instance. The CPU time in phase 2 was 3600 seconds.

5.6.2 Phase 1: Optimal Buffer Determination

So far we have taken the buffer parameter β as given input information, but the size of β has an immediate influence on the components of the objective function: When β is greater, both the waiting time and the deviation of the actual beam active time from the planned beam active time are smaller, yet the average beam active time generally is greater. We seek the optimal buffer parameter β^* for a given patient mix and the various objective function weights, which we use as input for the short-term optimization of the appointment schedule, given the actual data for a specific week.

To determine the optimal buffer parameter, we run a slightly different version of the optimization algorithm, in which the solution representation also contains the buffer parameter, which is initially randomly chosen between 0.5 and 0.99. A new individual in the GA inherits β from one of its parents. The mutation operator also modifies β of an individual by multiplying its current value by a random number between 0.75 and 1.25. To evaluate an individual's fitness during phase 1, we draw 1,000 random scenarios (STO evaluation strategy), simulate the actual starting times by applying the reactive procedure, and compare individuals according to the corresponding average objective function.

The determination of the optimal buffer parameter in phase 1 represents a strategic decision. Therefore, the algorithm running times are considerably extended. We run the modified version of the GA three, five, seven, and ten hours for the different instance sizes (i.e., 30, 50, 70, and 100 patients). Table 5.6 summarizes the findings for the previously noted average patient mix. For a given objective function weight of the waiting time λ_3, only small differences emerge among the different instance sizes. For each line, we find clear outliers in both the minimum

λ_3	P	mean	min	25%	50%	75%	max
0.1							
	30	0.69	0.52	0.68	0.70	0.71	0.74
	50	0.67	0.52	0.66	0.68	0.70	0.73
	70	0.67	0.50	0.64	0.68	0.70	0.75
	100	0.66	0.58	0.64	0.66	0.69	0.73
0.5							
	30	0.80	0.76	0.79	0.80	0.81	0.84
	50	0.79	0.74	0.79	0.80	0.80	0.83
	70	0.79	0.74	0.78	0.79	0.80	0.81
	100	0.78	0.71	0.77	0.79	0.79	0.82
1.0							
	30	0.83	0.78	0.82	0.83	0.84	0.86
	50	0.82	0.75	0.82	0.82	0.83	0.85
	70	0.81	0.75	0.80	0.82	0.82	0.85
	100	0.80	0.72	0.79	0.80	0.81	0.84

Table 5.6: Statistics of optimized buffer parameters β^* for 16 random instances with patient mix probabilities $\{43\%, 29\%, 22\%, 6\%\}$; 16 replications per instance.

and maximum optimized buffer parameter, which are also visible in the boxplot in Figure 5.10. Yet the inter-quartile range is generally small, leading us to conclude that the optimal buffer sizes are approximately 0.66 to 0.70 for $\lambda_3 = 0.1$, 0.78 to 0.80 for $\lambda_3 = 0.5$, and 0.80 to 0.83 for $\lambda_3 = 1.0$.

Figure 5.10 provides a boxplot of the optimized buffer parameters for the largest instance group with 100 patients. Once more we can see, that the optimal buffer size highly depends on the objective function weight. The greater the weight of the waiting time in the objective function, the higher the optimal buffer parameter will be. We also observe less variance in the optimal buffer parameter as λ_3 increases. This phenomenon is caused by our buffer definition, that is, as the percentile of the fitted distributions. The main driver of the objective function is beam active time, quantified in minutes. The larger the buffer already is, the greater the impact of a further increased buffer parameter on the planned activity duration in minutes and the total beam active time required. Take, for example, the distribution of the irradiation duration of patients in group 1. The median duration ($\beta = 50\%$) is 11.1. A buffer parameter of 60% would result in a planned duration of 12.2 minutes (+1.1 min), $\beta = 70\%$ lets the planned duration increase to 13.5 minutes (+1.3 minutes compared to 60% buffer parameter), and a β of 80% leads to a planned duration of 15.3 minutes (+1.8 minutes compared to 70% buffer parameter). For

larger buffer parameters, the planned durations and beam active time within the objective function thus are more sensitive to changes to β, leading to a more narrow boxplot for larger buffer parameters.

To investigate the dependence of the optimal buffer parameters on the patient mix, we also generated instances with arbitrary patient mixes. Again, each instance ran 16 times with different random seeds. Table 5.7 presents the average optimized buffer parameters for different patient mixes and instance sizes. The instances with solely patients from groups 3 ($\{0\%, 0\%, 100\%, 0\%\}$) and 4 ($\{0\%, 0\%, 0\%, 100\%\}$) – which feature the greatest variance and simultaneously the highest expected appointment durations – result in the lowest optimal buffers. Especially for patient group 4, beam active time increases drastically with a higher buffer percentile. This effect gets smaller as λ_3, the weight of waiting time in the objective function, increases and the importance of the beam active time simultaneously diminishes.

The optimized buffer parameters β^* of the other arbitrarily chosen patient mixes only slightly deviate from the optimized buffers that result from the patient mix we observed in the real-world data sets.

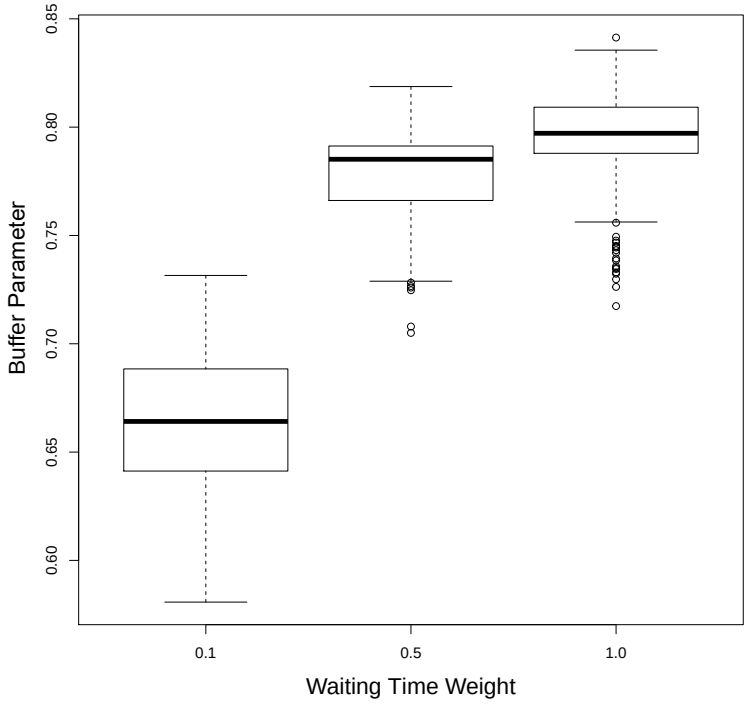


Figure 5.10: Boxplot of Optimal Buffer Parameters for Different Weights of Waiting Time in the Objective Function λ_3. Results from 16 randomly generated instances with 100 patients. The patient mix is chosen randomly according to the group probabilities $\{43\%, 29\%, 22\%, 6\%\}$. We use 16 genetic algorithm replications for each instance.

patient mix [%]	$\lambda_3 = 0.1$				$\lambda_3 = 0.5$				$\lambda_3 = 1.0$			
	30	50	70	100	30	50	70	100	30	50	70	100
$\{100, 0, 0, 0\}$	0.73	0.71	0.72	0.70	0.81	0.80	0.77	0.76	0.84	0.82	0.77	0.76
$\{0, 100, 0, 0\}$	0.71	0.72	0.70	0.72	0.82	0.80	0.80	0.79	0.84	0.84	0.82	0.82
$\{0, 0, 100, 0\}$	0.59	0.61	0.52	0.51	0.82	0.82	0.79	0.61	0.86	0.85	0.83	0.63
$\{0, 0, 0, 100\}$	0.54	0.61	0.59	0.58	0.70	0.72	0.69	0.62	0.77	0.77	0.70	0.61
$\{25, 25, 25, 25\}$	0.64	0.63	0.62	0.62	0.78	0.78	0.78	0.73	0.81	0.81	0.80	0.73
$\{40, 20, 20, 20\}$	0.69	0.59	0.69	0.64	0.79	0.79	0.76	0.73	0.83	0.82	0.78	0.74
$\{20, 40, 20, 20\}$	0.56	0.65	0.64	0.64	0.78	0.79	0.78	0.73	0.82	0.82	0.81	0.75
$\{20, 20, 40, 20\}$	0.60	0.59	0.51	0.61	0.79	0.79	0.76	0.69	0.82	0.82	0.80	0.70
$\{20, 20, 20, 40\}$	0.66	0.66	0.61	0.64	0.77	0.78	0.75	0.68	0.81	0.81	0.79	0.68

Table 5.7: Optimal buffer parameters of arbitrary patient mixes for instances including 30, 50, 70 and 100 patients. 16 replications per instance.

5.6.3 Phase 2: Schedule Optimization

The second phase of our computational study focuses on the comparison of the different solution evaluation approaches (DET, STO, WTE) from Section 5.5.4. We evaluate the resulting three baseline schedules on 1,000,000 random scenarios for the activity durations. Tables 5.8 to 5.10 list the average results for overall fitness (i.e., weighted objective), beam active duration, waiting times, and penalties for time window violations, across different objective function weights $\lambda_3 \in \{0.1, 0.5, 1.0\}$. We vary the buffer parameters β from 0.5 to 0.9 and include the optimal buffer parameters from phase 2. Some general patterns can be observed.

First, the waiting time is smallest for the stochastic optimization variant, as expected. The deterministic (DET) and waiting time estimation methods (WTE) indicate extremely large waiting times, especially when the buffer parameter β is small. The advantage of the stochastic optimization diminishes with an increasing buffer parameter β, as might be expected. The greater the buffer parameter, the longer the planned activity duration, and the smaller the probability that a patient will take longer than the planned duration. For a buffer parameter of $\beta = 0.50$, 50% (i.e., $1 - \beta$) of patients will not finish their treatment within the planned duration.

Second, the beam active time increases with a rising buffer parameter β yet the increase is astonishing small. Take for example the stochastic results of the 100 patients instances in Table 5.8. Increasing the buffer parameter from 0.50 to 0.60 leads to an increase in beam active time of 73 minutes (from 7980 to 8053), or 15 minutes per day on average. Simultaneously, the sum of all waiting times decreases by 1831 minutes (from 8570 to 6739), for a total reduction of 3.7 minutes per patient per day, given 100 patients and five days. Third, the penalty term is almost negligible for most instances. However, for very large buffer parameters, the total planned waiting time becomes so large that the time windows preferred by the patients cannot be respected anymore, leading to a significant penalty in the objective function. A buffer parameter of 0.90 or larger would only be optimal if the waiting time weight in the objective function were extremely high, which is not the case in practice.

When analyzing the behavior of the three solution approaches, we also can identify different patterns for the different buffer parameters β. First, for small β, the stochastic optimization algorithm has a clear advantage, in that it considers the waiting time of the patients directly by using sample average approximation. This effect gets stronger with a greater waiting time weight in the objective function (Tables 5.8 to 5.10).

		Av. Fitness			Av. Beam, $\lambda_1 = 1.0$			Av. Wait, $\lambda_3 = 0.1$			Av. Penalty, $\lambda_2 = 0.1$		
P	β	DET	STO	WTE	DET	STO	WTE	DET	STO	WTE	DET	STO	WTE
30	0.50	2778	**2694**	2781	2396	2453	2399	3817	2392	3812	6	16	7
30	0.60	2676	**2657**	2674	2419	2466	2419	2557	1904	2554	7	9	5
30	0.66	2633	**2635**	2631	2445	2474	2447	1872	1597	1839	10	9	7
30	0.68	2623	**2630**	2622	2459	2484	2459	1638	1443	1623	5	9	8
30	0.70	2619	2627	**2616**	2474	2497	2473	1439	1295	1425	9	10	8
30	0.80	**2678**	2700	2681	2610	2634	2612	657	626	649	27	28	33
30	0.90	**3049**	3077	3051	2996	3025	2999	317	305	314	214	208	208
50	0.50	4709	**4315**	4704	3809	3890	3808	8984	4200	8934	19	57	18
50	0.60	4396	**4259**	4391	3827	3919	3831	5660	3348	5575	32	49	27
50	0.66	4247	**4228**	4248	3861	3940	3867	3814	2818	3767	53	62	43
50	0.68	4221	4220	**4216**	3888	3956	3890	3275	2571	3210	54	69	49
50	0.70	4195	4212	**4192**	3914	3978	3915	2750	2266	2717	61	80	60
50	0.80	**4273**	4320	4276	4152	4204	4156	1057	1009	1047	155	150	149
50	0.90	**4908**	4988	4917	4794	4873	4802	475	450	471	674	693	672
70	0.50	7125	**6146**	7089	5427	5534	5452	16968	5983	16358	6	133	7
70	0.60	6463	**6049**	6449	5449	5578	5465	10120	4631	9825	16	81	18
70	0.66	6156	**6020**	6144	5512	5615	5511	6408	4000	6306	24	52	23
70	0.68	6082	**6022**	6067	5544	5640	5542	5341	3767	5211	35	50	33
70	0.70	6023	**6009**	6013	5579	5664	5579	4402	3396	4309	37	57	38
70	0.80	**6127**	6198	6134	5964	6040	5973	1468	1398	1453	158	177	163
70	0.90	**7137**	7298	7150	6908	7048	6920	628	588	623	1661	1912	1671
100	0.50	10878	**8860**	10681	7891	7980	7940	29865	8570	27398	2	227	5
100	0.60	9574	**8736**	9498	7863	8053	7901	17108	6739	15964	10	89	10
100	0.66	8965	**8648**	8944	7926	8086	7952	10374	5570	9904	14	54	14
100	0.68	8825	**8671**	8815	7990	8139	8013	8327	5282	8000	19	43	18
100	0.70	8722	**8664**	8711	8055	8181	8068	6644	4776	6403	27	52	32
100	0.80	**8883**	8975	8892	8646	8733	8658	2032	1951	2012	345	461	322
100	0.90	**10624**	10939	10645	10002	10169	10019	843	798	838	5370	6905	5421

Table 5.8: Average results of 16 randomly generated instances and 16 replications for each instance. $\lambda_3 = 0.1$, optimization time limit of one hour. Bold numbers represent the best solutions per line. Lines with optimal buffer parameters are in gray.

		Av. Fitness			Av. Beam, $\lambda_1 = 1.0$			Av. Wait, $\lambda_3 = 0.1$			Av. Penalty, $\lambda_2 = 0.5$		
P	β	DET	STO	WTE	DET	STO	WTE	DET	STO	WTE	DET	STO	WTE
30	0.50	4300	**3447**	4243	2400	2572	2483	3800	1740	3520	0	11	1
30	0.60	3700	**3252**	3679	2419	2591	2428	2562	1317	2501	0	5	0
30	0.70	3198	**3081**	3184	2478	2599	2481	1440	960	1405	1	4	0
30	0.78	2968	2974	**2964**	2580	2633	2580	774	675	765	3	7	4
30	0.80	2946	2958	**2941**	2615	2657	2615	654	593	645	6	9	6
30	0.83	2960	2975	**2955**	2697	2727	2697	512	479	501	15	17	15
30	0.90	**3237**	3265	3240	3019	3049	3022	312	295	307	125	138	128
50	0.50	8289	**5653**	7928	3819	4052	4219	8939	3138	7411	3	64	7
50	0.60	6670	**5272**	6536	3835	4102	4043	5663	2299	4979	7	41	8
50	0.70	5305	**4979**	5266	3926	4136	3953	2726	1633	2595	32	53	31
50	0.78	4785	4807	**4774**	4113	4218	4119	1262	1086	1227	82	93	82
50	0.80	4747	4788	**4743**	4168	4252	4172	1051	953	1035	108	119	107
50	0.83	4778	4830	**4776**	4298	4373	4301	803	748	790	156	165	160
50	0.90	**5320**	5407	5323	4833	4917	4844	467	438	460	507	543	499
70	0.50	13890	**8214**	12405	5438	5753	6007	16903	4764	12779	1	158	18
70	0.60	10505	**7521**	9857	5467	5834	5913	10071	3283	7878	5	92	9
70	0.70	7778	**7091**	7679	5594	5900	5666	4351	2336	4010	16	45	15
70	0.78	6827	6864	**6808**	5911	6060	5918	1769	1520	1721	65	86	59
70	0.80	6763	6825	**6756**	5990	6112	6002	1454	1315	1419	92	112	90
70	0.83	6810	6902	**6809**	6191	6302	6199	1087	1012	1066	151	188	156
70	0.90	**7931**	8245	7947	6966	7104	6975	613	571	608	1316	1710	1336
100	0.50	22637	**11922**	19351	7899	8275	8886	29475	7042	20898	0	252	32
100	0.60	16295	**10963**	14715	7888	8400	8730	16812	5030	11956	1	96	15
100	0.70	11363	**10220**	11134	8108	8506	8325	6508	3395	5614	3	34	3
100	0.78	9850	9928	**9846**	8580	8754	8609	2450	2135	2366	90	213	109
100	0.80	9793	9952	**9778**	8690	8854	8709	2010	1823	1961	196	372	179
100	0.83	9953	10140	**9948**	8995	9136	9013	1481	1395	1451	436	613	420
100	0.90	**12937**	13796	12961	10070	10241	10087	830	785	822	4904	6326	4927

Table 5.9: Average results of 16 randomly generated instances and 16 replications for each instance. $\lambda_3 = 0.5$, optimization time limit of one hour. Bold numbers represent the best solutions per line. Lines with optimal buffer parameters are in gray.

		Av. Fitness			Av. Beam, $\lambda_1 = 1.0$			Av. Wait, $\lambda_3 = 0.1$			Av. Penalty, $\lambda_2 = 1.0$		
P	β	DET	STO	WTE	DET	STO	WTE	DET	STO	WTE	DET	STO	WTE
30	0.50	6209	**4265**	5866	2399	2616	2696	3809	1636	3167	0	13	2
30	0.60	4983	**3874**	4838	2422	2654	2660	2561	1211	2177	0	9	1
30	0.70	3915	**3509**	3889	2478	2679	2557	1437	826	1332	0	4	0
30	0.78	3356	**3296**	3338	2580	2696	2583	774	596	753	2	4	2
30	0.80	3275	**3252**	3262	2616	2701	2618	654	545	639	5	7	5
30	0.83	3220	3220	**3212**	2698	2756	2700	510	450	498	12	15	14
30	0.90	3456	3483	**3451**	3026	3068	3029	310	284	304	120	130	118
50	0.50	12743	**7207**	11239	3815	4109	4484	8927	3031	6746	1	68	9
50	0.60	9520	**6376**	8617	3846	4186	4455	5670	2141	4151	5	49	11
50	0.70	6684	**5716**	6529	3929	4247	4176	2725	1423	2323	29	46	30
50	0.78	5453	**5374**	5412	4115	4300	4137	1258	984	1195	80	90	80
50	0.80	5327	5306	**5298**	4178	4310	4185	1042	886	1009	106	110	105
50	0.83	5263	5296	**5250**	4304	4416	4316	801	715	775	158	166	158
50	0.90	5803	5895	**5789**	4851	4938	4852	462	428	456	490	529	481
70	0.50	22369	**10605**	18078	5439	5813	6376	16928	4619	11664	1	173	38
70	0.60	15490	**9150**	13232	5467	5925	6341	10020	3132	6872	4	93	19
70	0.70	9941	**8185**	9531	5602	6037	6106	4325	2096	3410	14	51	15
70	0.78	7729	**7632**	7682	5918	6147	5949	1757	1409	1676	55	76	57
70	0.80	7529	7530	**7488**	5999	6192	6017	1448	1229	1392	83	110	79
70	0.83	7419	7508	**7398**	6199	6355	6210	1082	975	1050	138	178	138
70	0.90	**8861**	9334	8884	6978	7129	6998	611	563	601	1273	1642	1286
100	0.50	37421	**15382**	28795	7899	8372	9071	29522	6778	19666	0	232	58
100	0.60	24600	**13428**	20141	7898	8532	9078	16702	4782	11031	0	113	32
100	0.70	14576	**11801**	13699	8123	8681	8844	6451	3083	4849	2	37	6
100	0.78	11125	**11055**	11024	8594	8887	8660	2440	1971	2281	91	197	82
100	0.80	10880	10989	**10830**	8712	8946	8746	1989	1727	1905	179	316	180
100	0.83	10871	11142	**10852**	9019	9193	9035	1469	1357	1428	384	592	389
100	0.90	**15607**	17397	15783	10088	10275	10111	828	780	819	4692	6342	4853

Table 5.10: Average results of 16 randomly generated instances and 16 replications for each instance. $\lambda_3 = 1.0$, optimization time limit of one hour. Bold numbers represent the best solutions per line. Lines with optimal buffer parameters are in gray.

To assess the quality of the three approaches, we also performed statistical tests, namely, Wilcoxon-Mann-Whitney (WMW) tests on combinations of patient size, buffers, objective function weights, and instances. The sample size per test for each method (DET, STO and WTE) is 16 (different seeds), and we performed pairwise tests. We chose a significance level of $\alpha = 0.05$. Table 5.11 shows the results of the significance tests for 100 patients. The values in the table indicate the number of instances in which one of the methods is better, or else the tests reveal no significant difference among methods.

For the comparison of STO, and DET, the statistical tests confirm the previously identified trend: For small β, STO is clearly better than DET, but as β increases, it becomes more beneficial to use the deterministic approach. A similar result emerges from the comparison of STO and WTE over all λ_3. Comparing WTE to DET we conclude, that on average, WTE gives better results than DET for small values of β. Hence, if the scenario approach is computationally too expensive, the waiting time estimation approach should be favored. For large values of β the statistical tests indicate only for a few instances a difference between the results of DET and WTE.

Using the buffer parameter sizes optimized in phase 2, the picture changes slightly, and all solution approaches lead to good results. A small advantage accrues to the WTE approach, which performs slightly better than DET on average, followed by STO. However, the differences among the three approaches are rarely statistically significant (Wilcoxon-Mann-Whitney Test) for the optimized buffer sizes. Finally, a higher than optimal β favors DET over all other approaches (though not statistically significantly, as Table 5.11 shows). Note, that although STO usually does not provide superior results for operational (weekly) planning problems for optimized buffer sizes relative to the two other approaches DET and WTE, it is required to solve the strategic problem of determining the optimal buffer parameter itself. For this purpose, it cannot be replaced by the other approaches.

Figure 5.11 shows the evolution of waiting times and beam active times, depending on the buffer parameter β. Again, we see that the different solution approaches DET, STO, and WTE differ substantially in the waiting times for small β. The larger the buffer parameter, the more similar the results.

		STO vs. DET			STO vs. WTE			WTE vs. DET		
λ_3	β	STO	DET	equal	STO	WTE	equal	WTE	DET	equal
0.1										
	0.50	16	0	0	16	0	0	12	0	4
	0.60	16	0	0	16	0	0	7	0	9
	0.66	16	0	0	16	0	0	2	0	14
	0.68	15	0	1	14	0	2	1	0	15
	0.70	10	0	6	10	0	6	2	0	14
	0.78	0	13	3	0	11	5	0	1	15
	0.80	0	13	3	0	13	3	0	2	14
	0.83	0	15	1	0	14	2	0	0	16
	0.90	0	16	0	0	16	0	0	2	14
0.5										
	0.50	16	0	0	16	0	0	12	0	4
	0.60	16	0	0	16	0	0	14	0	2
	0.66	16	0	0	16	0	0	13	0	3
	0.68	16	0	0	16	0	0	13	0	3
	0.70	16	0	0	16	0	0	11	0	5
	0.78	1	5	10	1	9	6	4	1	11
	0.80	0	14	2	0	14	2	1	1	14
	0.83	0	16	0	0	16	0	0	0	16
	0.90	0	16	0	0	16	0	0	0	16
1.0										
	0.50	16	0	0	16	0	0	15	0	1
	0.60	16	0	0	16	0	0	15	0	1
	0.66	16	0	0	16	0	0	16	0	0
	0.68	16	0	0	16	0	0	15	0	1
	0.70	16	0	0	16	0	0	14	0	2
	0.78	14	2	0	10	2	4	12	0	4
	0.80	8	4	4	1	8	7	12	0	4
	0.83	0	12	4	0	16	0	8	0	8
	0.90	0	16	0	0	16	0	0	1	15

Table 5.11: Wilcoxon-Mann-Whitney test to pairwise compare methods STO, DET and WTE for 16 random instances with 100 patients. 16 replications per run. Significance level is $\alpha = 0.05$. Entries show – for different λ_3 and β – the number of instances where either of the two pairwise compared methods is significantly better than the other method. Column "equal" lists the number of instances, where neither of the methods performed better than the other. Lines with optimal buffer parameters in gray.

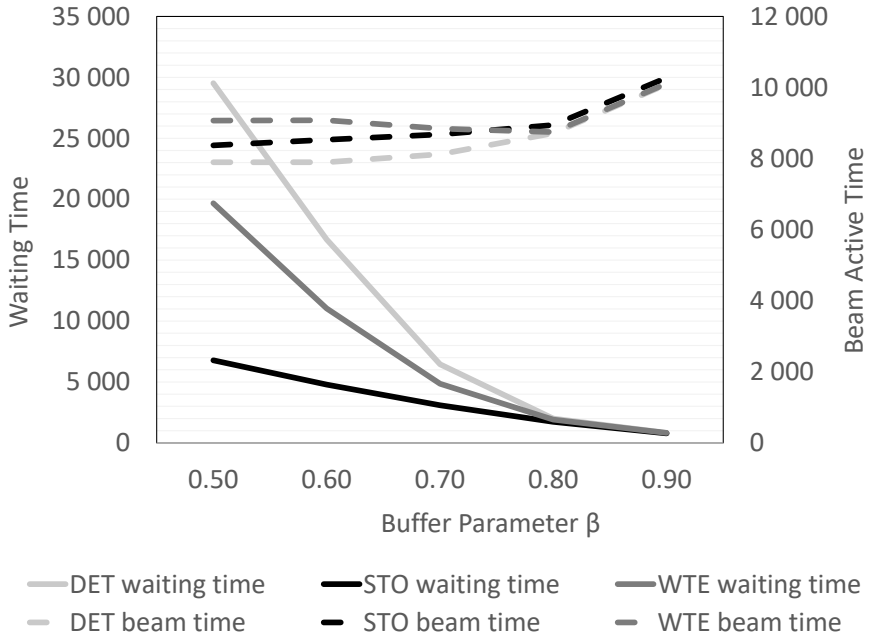


Figure 5.11: Average Results for 100 Patients from Table 5.10, Graphically.

5.7 Summary

We reconfirm with this chapter that appointment duration is highly stochastic, even in the case of radiotherapy, where the treatment and its duration are planned in detail beforehand. The best fitting distribution relies on the Burr and Dagum families of solutions for the pre-/post-treatment phase and for the treatment itself, respectively. To deal with this stochasticity, we modified an algorithm that has been applied to the deterministic variant of the highly constrained problem of radiotherapy appointment scheduling. To account for possible variations in appointment durations, we introduce a buffer parameter, the percentile of the fitted distribution, to extend the planned activity durations.

Depending on the weight λ_3 of patient waiting time, relative to the beam active time weight λ_1, the optimal buffer percentile of the underlying distribution function varies. The higher waiting time is weighted in the objective function, the higher the optimal buffer parameter is. For example, the balanced scenario with $\lambda_1 = \lambda_3 = 1.0$ requires $\beta^* \in [0.80, 0.83]$ while for $\lambda_1 = 1.0$ and $\lambda_3 = 0.1$, $\beta^* \in [0.66, 0.68]$. The strategic decision about how much buffer should be added to the initially planned (or median) appointment duration is therefore highly dependent on the importance that management assigns to patient waiting time relative to machine usage.

The determination of the optimal buffer size itself demands stochastic sampling. However, we also compare different options for evaluating a solution during the

optimization procedure to the stochastic method STO, namely the deterministic variant DET that completely ignores waiting time during the optimization and a quasi-deterministic variant WTE, which estimates the expected waiting time of a schedule through regression. We highlight the importance of stochastic sampling techniques, especially when the buffer parameter is smaller than optimal. For example, in case $\beta = 0.5$ and $\lambda_3 = 0.1$, the deterministic algorithm leads to approximately four times the patient waiting time than the stochastic counterpart. Furthermore, even a slight increase in the planned activity duration, through an increase in the buffer parameter, can reduce patient waiting time drastically and causes only a minor increase in beam active time. For example, increasing the buffer parameter from $\beta = 0.5$ to $\beta = 0.6$ scales down patient waiting time by 20-30% and causes an increase in machine uptime of only 1-2% in the case of stochastic optimization. By investing a little extra time on the resources, the facility can exert an enormous impact on patient's waiting time and satisfaction. We also show, that for the optimal buffer size, the advantage of the stochastic optimization diminishes and the WTE method is favored.

Chapter 6

Conclusions to Part I

Scheduling recurring radiotherapy appointments is a complex problem, which can only be solved by hand up to a very limited size. Hence, optimization algorithms are necessary to achieve good quality solutions, which typically strive to optimize the usage of the bottleneck resource, which is the particle beam. The specific problem dealt with in this thesis occurs in ion beam facilities, which are equipped with only one ion beam which can be directed to multiple treatment rooms. Additional constraints such as optional activities or alternative resources further complicate the optimization problem.

We have solved the long-term radiotherapy appointment scheduling problem deterministically, considering – besides the already mentioned constraints – so-called "stable starting times", meaning that we assigned treatment start times to the same patient which only vary slightly from day to day. We have shown, that solving this type of scheduling problem exactly is intractable already for small instances. Therefore, we address the optimization problem by developing multiple alternative heuristical methods tailored to the problem at hand. We conclude, that a simple combination of population-based and individual-based heuristics performs best for most instances.

However, in realizing that appointment durations are highly stochastic, we aim for a solution method, which – besides maximizing beam usage – also focuses on patient waiting time, which tends to accumulate from patient to patient on one day. To do so, we thoroughly analyzed data gathered at an ion beam center in Wiener Neustadt/Austria and fitted theoretical distributions to the data. Then, we define the βth percentile of those distributions as a buffer parameter to the optimization, where we use the corresponding percentile activity durations instead of the expected durations when creating a baseline schedule. As an optimization algorithm, we use the genetic algorithm proven to be successful in the deterministic setting and apply three different strategies to evaluate the quality of intermediate solutions (i.e., an estimate of the *actual* beam usage and the *actual* patient waiting time) during the

optimization procedure: While the deterministic estimation strategy ignores patient waiting time when building a baseline schedule, the stochastic estimation variant draws random scenarios from the fitted distributions and evaluates the quality of a solution according to the average scenario solution, using a reactive procedure which mimics the human decision maker every time the actual schedule deviates from the baseline (i.e., the planned) schedule. Additionally, we have presented an estimation based evaluation strategy, which builds upon the deterministic variant, but benefits from the correlation between beam idle time and patient waiting time.

Computational results show, that the optimal buffer parameter in the latter problem highly depends on the weight of patient waiting time compared to the beam usage weight in the objective function. Nevertheless, for any objective function weight, already a small increase in the buffer parameter from e.g. $\beta = 0.5$ to $\beta = 0.6$ has a tremendous effect on patient waiting time while beam usage remains almost unchanged. For small (i.e., lower than optimal) buffer parameters, it is also highly recommended to follow the stochastic evaluation scheme, as this results in dramatically lower patient waiting time but comparably low beam usage in contrast to the deterministic variant. Additionally, the waiting time estimation strategy is favored over the deterministic setting for many optimal buffer sizes β^*. We conclude, that even though patient waiting time is not yet on the optimization agenda of many radiotherapy centers, we recommend to include a comparable figure into the objective function when building baseline schedules, such as to increase patient well-being and satisfaction during the fight against cancer.

Part II

Integrated Activity Selection and Staff Scheduling at the Red Cross Blood Donation Services

Chapter 7

Integrated Activity Selection and Skilled Staff Scheduling

The second project of the PhD-thesis deals with scheduling doctors, nurses and medical auxiliary personnel working for the blood donation department of the Austrian Red Cross in Vienna, Lower Austria and Burgenland. While the Red Cross runs a fixed blood donation center in Vienna with regular opening hours from Monday to Friday, the vast majority of blood donations is collected during so-called mobile blood donation operations (80%). For these operations, skilled staff is needed for different work stages of the blood collection process. The staff scheduling problem is constrained by maximum working hours per week and per day as well as rest times required by law. Furthermore, employees offer different skills. The costs involved with each working hour, however, might not only depend on the skills of the employee but also on his/her contract type. Additionally, employees have a special agreement concerning available weekdays. For example, an employee might only be available from Monday to Friday, while another employee might only work on weekends. Finally, since the blood donation operations could take place also in the rural area of Austria, driving from the home base in Vienna to the blood donation location and back also adds up to the working hours.

Additional degree of freedom is added to the model by allowing for flexible blood donation operation dates: For example, one operation should take place between 3rd and 5th of January, but the exact date is flexible and should be fixed during the optimization procedure. This could further improve the quality of the solution since we could then circumvent some of the hard constraints of the model such as minimum rest time if we postponed/preponed the operation.

The upcoming chapters and sections address this real-world problem scientifically.

7.1 Introduction

Staff scheduling or rostering is one of the most difficult planning problems companies face (De Bruecker et al. [2015]) and is specifically important for companies operating in either high-wage countries and/or the service sector, which typically faces high labor intensity on the one hand and cost pressure on the other hand (Felberbauer et al. [2018], Heimerl and Kolisch [2010]). Additionally, staff preferences become more and more important and are reflected by different contract types, working hour agreements and availabilities. Staff members might additionally possess different skills, typically leading to a highly heterogeneous set of employees (De Bruecker et al. [2015]). Working time regulations and labor law further increase the complexity of staff scheduling problems. For example, between two working days a minimum rest time needs to be assured or a maximum number of consecutive working days needs adhere (Van Den Bergh et al. [2013]). This complex planning problem has been widely studied in the literature, mostly taking the staff demand on each day as given. But as Heimerl and Kolisch [2010] state, the staffing decision is only the last of multiple interleaved planning phases. Typically the resource requirements are fixed before the assignment of human resources to work packages is done, leading to possibly sub-optimal solutions.

In this study, we address a real-world inspired combination of activity selection, activity scheduling, and corresponding staff assignment to serve the selected activities. A comparable planning problem arises at the Austrian Red Cross Blood Donation Services, for example. We strive to minimize internal and potentially external labor costs. The key questions include which of the potential activities should be operated on which days and which human resources should be assigned to those activity-day combinations.

This part of the thesis is organized as follows: Section 7.2 gives insight into related work on employee scheduling and activity selection. Section 7.3 presents the formal problem statement of the integrated activity selection and staff scheduling problem. Section 7.4 is devoted to the mathematical programming formulation of the proposed problem. In Section 8.1 we present a hierarchical solution approach. Section 8.2 is dedicated to intensive computational tests of different variants of the presented solution approach as well as a comparison to exact solvers. Finally, Section 9 concludes this part and proposes some possible future research extensions.

7.2 Related Work

Employee scheduling problems are studied intensively in the literature and multiple review papers are available on this topic: Van Den Bergh et al. [2013] give a general overview of constraints and solution methods to this category of optimization problems and in total summarize more than 300 papers in this field. They list time-related constraints like a maximum number of consecutive working days or a minimum number of consecutive days off, many of which are also relevant for the staffing decision in this paper. The second review by De Bruecker et al. [2015] focuses specifically on workforce planning problems that incorporate skills. They list multiple application areas of skilled staff scheduling problems, ranging from health care to maintenance, call center, and transportation services as well as classical production and manufacturing planning. The nurse rostering problem as a special type of workforce planning problem is categorized in De Causmaecker and Van Den Berghe [2011]. Among the many interesting papers listed in those review articles, we would like to mention a view in more detail (without this list being exhaustive).

Bard and Purnomo [2005] propose a preference based nurse scheduling problem and strive for elaborating more flexible arrangements for working hours and working days by softening the classical shift structure. They try to accommodate individual nurse preferences in terms of specific days off requests or general weekly working times. Naudin et al. [2012] present three mathematical models to solve a highly constrained staff rostering problem, where understaffing is generally allowed but penalized in the objective function. They consider weekly work duration, day-off, and minimum daily rest duration constraints, among others. Firat and Hurkens [2012] provide a mixed integer programming (MIP) based approach for the multi-skill workforce scheduling problem, in which they assign tasks to technician teams. They consider availabilities of technicians as well as outsourcing possibilities but assume skills to be hierarchical. They propose a flexible matching model which works especially well for rare skills. In a more recent paper, Dahmen and Rekik [2015] develop and present a hybrid heuristic which combines branch-and-bound and tabu search to solve a multi-activity, multi-day shift scheduling problem.

An interesting application area of staff scheduling is crew rostering in the airline industry, which is addressed by Maenhout and Vanhoucke [2010], for example. The authors strive for minimizing operational costs, consisting of both a combination of overtime and external staff costs and maximizing the social quality of the schedules, where crew members may express preferred working days. Maenhout and Vanhoucke [2010] present multiple different metaheuristic algorithms and improvement principles and compare their results with results obtained by exact MIP solvers.

The combination of scheduling and staffing of projects is addressed in Heimerl

and Kolisch [2010], who propose a mixed integer programming model to solve their research problem and compare the results with simple heuristics. They show, that in their specific formulation, commercial MIP solvers like CPLEX can find a (near) optimal solution within a few seconds and investigate the influence of parameter changes on personnel costs. In Kolisch and Heimerl [2012] the authors develop a hybrid metaheuristic approach to address the same problem, which is efficient also for larger problem sizes. Felberbauer et al. [2018] build upon those results but also consider stochastic work package processing times. They decompose the problem into a project scheduling and a staffing subproblem, solving the first subproblem using an iterated local search heuristic. A similar approach is followed in Gutjahr and Froeschl [2013], who decompose their problem into portfolio determination and staffing. Another way to address the combination of task scheduling and employee assignment is suggested by Drezet and Billaut [2008], who consider staff resources in a project scheduling environment but still respect skills and legal labor constraints during their optimization. Their focus, however, is to minimize maximal lateness of projects and they solve their problem using tabu search. More recently, Smet et al. [2016] present an integrated task and employee scheduling problem with multi-skilled workforce where tasks need to be assigned to shifts and then employees are assigned to the task-shift combinations. The authors propose various constructive heuristics and a very large neighborhood search heuristic to address the two highly intertwined subproblems.

7.3 Problem Statement

The current problem is motivated by the specific planning problem blood donation organizations face. In Austria, blood is donated by volunteers in the field. A blood donation team consisting of doctors, nurses, and medical assistants travels to an agreed location and welcomes participants from the surrounding area. For example, such a location might be a school, university, city hall or a company inviting the blood donation service to conduct the blood donation activity in their premises. Furthermore, the Austrian blood donation service is part of the Austrian Red Cross, which across the country has its own facilities which are mainly used for ambulance services but are also home to blood donation activities on a regular basis.

We are given a set of jobs (or in our terminology activities) with known daily start and end times, duration, employee requirements in terms of skills, a set possible execution days, and a reward. Among the set of eligible activities, we select a subset of activities to take place, such that a minimum reward is achieved. Typically, the set of eligible activities is considerably larger than the set of chosen activities, resulting in extensive flexibility in the activity selection phase.

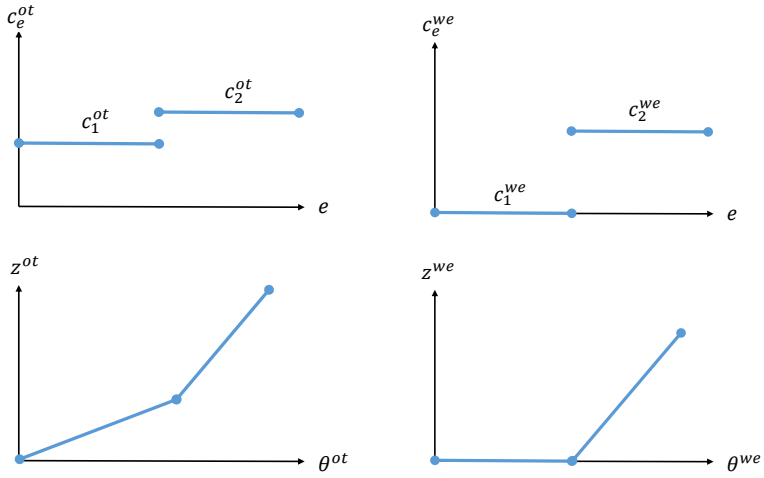
For each chosen activity, we furthermore select a realization date among the

set of eligible days per activity. Each activity can only take place on one of the possible days, non-chosen activities do not have an assigned date. We call the results of these two decisions the "activity-day-combination".

The required skills of all activities taking place are preferably fulfilled by contracted employees, which are summarized in a set of internal employees. Each employee has signed a contract with the company, indicating available weekdays from Monday to Sunday, contractual working hours per week and month, minimum and maximum working hours per day as well as acquired non-hierarchical skills. In our specific problem setting, we deal with seven different skills and three profession groups, namely doctors, nurses, and medical assistants. Doctors exclusively have a minimum of one out of two possible skills, nurses have one or two specific skills that doctors do not have and finally, medical assistants might have a combination of up to three exclusive skills, that neither doctors nor nurses possess.

When assigning an employee to a set of activities that should be served, the following constraints and regulations need to be respected:

- Employees can only work on one activity per day.
- Employees can only be assigned to activities that meet their minimum/ maximum working hours per day.
- Employees can only be assigned to activities taking place on their available weekdays.
- Employees can only work in acquired skills.
- Between two consecutive days, employees must have a minimum rest time of 11 hours. Hence, an activity finishing at 9 pm on day 1 and an activity starting at 7 am on day 2 cannot be served by the same employee, as the rest time would only be 10 hours.
- Employees must have two consecutive days off per week (where one week is assumed to be the time span from Monday to Sunday).
- Employees can work on maximum nine consecutive days.
- Employees who do not have a specific "weekend contract" must have one weekend off per month (a month in our case is equivalent to a time span of 4 consecutive weeks, which also is the planning horizon of our problem).
- Regular working time is limited by the employee's contract. Employees can work overtime though, however, overtime is limited to 25% of the employee's regular working time per month.



(a) Overtime (b) Weekend

Figure 7.1: Step Functions of Overtime and Weekend Costs for a given Profession, e.g., Doctors.

In choosing the activity-day-combinations and assigning employees to activities, we strive to minimize costs arising from overtime hours worked, weekend hours worked (by employees who cause costs when working on Saturdays or Sundays), and penalties arising from unserved positions, in case required skills for a specific activity are not met. Here, we consider for each employee e specific overtime costs per overtime hour worked and weekend costs per weekend hour worked, c_e^{ot} and c_e^{we}, respectively. In our setting, for each profession, there are two cost rates, one for employees who signed their labor contract before an amendment took place receiving a higher wage for overtime and weekend hours worked and one for employees who signed their contract past those modifications. We will call these contract types "high" and "standard" contracts. When sorting the employees according to their overtime and weekend costs, c_e^{ot} and c_e^{we} form a step function, as depicted in Figure 7.1. θ^{ot} and θ^{we} describe the number of overtime and weekend hours worked, respectively. Then, the bottom part of Figure 7.1 depicts the approximate increase in overtime and weekend costs for higher required overtime and weekend hours, respectively. The costs per profession and contract type ("standard" and "high") are summarized in Table 7.1. Note, that these figures do not reflect real personnel costs due to privacy reasons. Hence, we distorted the real-world cost factors using a constant factor.

	Overtime, c_e^{ot}		Weekend, c_e^{we}	
Profession	high	standard	high	standard
Doctor	10.35	6.45	6.90	0.0
Nurse	4.35	3.45	2.90	0.0
Med. Ass.	3.15	2.85	2.10	0.0

Table 7.1: Cost Factors c_e^{ot} and c_e^{we} for the Different Professions.

Sets	
Notation	Description
$\mathcal{J}$	Set of jobs (= blood donation activities), indices $j \in \{1, ..., J\}$.
$\mathcal{E}$	Set of employees, indices $e \in \{1, ..., E\}$.
$\mathcal{E}'$	Subset of employees who do *not* have a weekend contract.
$\mathcal{D}$	Set of days, indices $d \in \{1, ..., D\}$.
$\mathcal{D}^{Mo}, \mathcal{D}^{Sa}$	Subsets of weekdays on Mondays and Saturdays, respectively.
$\mathcal{D}^{we}$	Subset of weekend-days.
$\mathcal{D}^j$	Subset of possible days for activity j.
$\mathcal{J}^d$	Subset of possible activities on working day d.
$\mathcal{S}$	Set of skills, indices $s \in \{1, ..., S\}$.
$\mathcal{P}$	Set of professions $p \in \{1, ..., P\}$.
$\mathcal{E}^p$	Set of employees belonging to profession p.
$\mathcal{S}^p$	Set of skills belonging to profession p.

Table 7.2: Sets of the Mathematical Modeling Formulation.

7.4 Problem Formulation

The objective function and constraints described in Section 7.3 can be formulated mathematically. We denote activities by index $j \in \mathcal{J}$; employees are in set $e \in \mathcal{E}$. Table 7.2 lists the symbols and sets used in the formulation of the problem. All necessary input information is summarized in Table 7.3. The main decision variable of the mathematical model is x_{ejsd}, describing whether employee e is assigned to perform skill $s \in \mathcal{S}$ in activity j on day $d \in \mathcal{D}^j$; the latter set consists of all eligible days for activity j. Table 7.4 gives an overview of all decision variables of the mathematical modeling formulation.

$$\text{minimize} \sum_{e \in \mathcal{E}} u_e \cdot c_e^{ot} + \sum_{e \in \mathcal{E}} \sum_{d \in \mathcal{D}^{we}} z_{ed} \cdot c_e^{we} + \sum_{j \in \mathcal{J}} \sum_{s \in \mathcal{S}} p_{js} \cdot Pen \tag{7.4.1}$$

Input Parameters	
Notation	Description
c_e^{we}	Costs arising for each hour employee e works on the weekend (i.e., on Saturday or Sunday).
h_e^m	Contractual working hours for employee e (4 weeks).
c_e^{ot}	Costs arising for each overtime hour employee e works, i.e., the hours exceeding h_e^m.
a_{ejd}	Binary variable, indicating whether employee e can work at activity j if it takes place on day d due to his contract (weekday availability, duration constraints).
c_{es}	Binary variable, indicating whether employee e has skill s ("capability").
$f_{jdj'd'}$	Binary variable, indicating whether both activities j and j' can be served by the same employee (due to rest time constraints) if they take place on days d and d', respectively.
r_{js}	Resource requirements of skill s at activity j.
t_j	Duration of activity j.
b_j	Expected reward (number of blood donations) at activity j.
B^{min}	Minimum reward (number of blood donations) needed during the planning horizon.
Pen	Penalty of external staff.
M	Very large number.

Table 7.3: Input Parameters to the Mathematical Modeling Formulation.

Decision Variables	
Notation	Description
x_{ejsd}	Binary variable indicating whether employee e is working at activity j in skill s on day d.
p_{js}	Variable indicating the number of skills of activity j that are fulfilled using external staff.
y_{ed}	Binary variable indicating whether worker e is active on day d.
w_{ed}	Binary variable indicating whether worker e is active on day pair $(d,\, d+1)$.
z_{ed}	Total working time of employee e on day d.
u_e	Overtime hours worked by employee e.
v_j	Binary variable indicating, whether activity j is taking place or not.
φ_{jd}	Binary variable indicating, whether activity j takes place on day d or not.

Table 7.4: Variables of the Mathematical Modeling Formulation.

Subject to:

$$\sum_{j\in\mathcal{J}^d}\sum_{s\in\mathcal{S}} x_{ejsd} = y_{ed} \quad \forall e\in\mathcal{E}, d\in\mathcal{D} \tag{7.4.2}$$

$$x_{ejsd} \le y_{ed} \quad \forall e\in\mathcal{E}, j\in\mathcal{J}, s\in\mathcal{S}, d\in\mathcal{D}^j \tag{7.4.3}$$

$$w_{ed} \ge y_{ed} \quad \forall e\in\mathcal{E}, d\in\mathcal{D} \tag{7.4.4}$$

$$w_{ed} \ge y_{e(d+1)} \quad \forall e\in\mathcal{E}, d\in\mathcal{D}\setminus\{D\} \tag{7.4.5}$$

$$w_{ed} \le y_{ed} + y_{e(d+1)} \quad \forall e\in\mathcal{E}, d\in\mathcal{D}\setminus\{D\} \tag{7.4.6}$$

$$\sum_{e\in\mathcal{E}}\sum_{d\in\mathcal{D}^j} x_{ejsd} + p_{js} \ge r_{js}\cdot v_j \quad \forall j\in\mathcal{J}, s\in\mathcal{S} \tag{7.4.7}$$

$$\sum_{s\in\mathcal{S}} x_{ejsd} \le a_{ejd} \quad \forall e\in\mathcal{E}, j\in\mathcal{J}, d\in\mathcal{D}^j \tag{7.4.8}$$

$$\sum_{j\in\mathcal{J}}\sum_{d\in\mathcal{D}^j} x_{ejsd} \le c_{es}\cdot M \quad \forall e\in\mathcal{E}, s\in\mathcal{S} \tag{7.4.9}$$

$$\sum_{s} x_{ejsd} + \sum_{s} x_{ej'sd} \le 1 \quad \forall e, j, j'\ne j, d\in\mathcal{D}^j, d\in\mathcal{D}^{j'} \tag{7.4.10}$$

$$\sum_{s} x_{ejsd} + \sum_{s} x_{ej'sd'} \le f_{jj'} + 1 \quad \forall e, j, j'\ne j, d\in\mathcal{D}^j, d'\in\mathcal{D}^{j'} \tag{7.4.11}$$

$$\sum_{d=d'}^{d'+5} w_{ed} \le 5 \quad \forall e\in\mathcal{E}, d'\in\mathcal{D}^{Mo} \tag{7.4.12}$$

$$\sum_{d\in\mathcal{D}^{Sa}} w_{ed} \le 3 \quad \forall e\in\mathcal{E}' \tag{7.4.13}$$

$$\sum_{d=d'}^{d'+F_{max}} y_{ed} \leq F_{max} \quad \forall e \in \mathcal{E}, d' \in \mathcal{D} \setminus \{D - F_{max} - 1, ..., D\} \tag{7.4.14}$$

$$z_{ed} = \sum_{j \in \mathcal{J}^d} \sum_{s \in \mathcal{S}} t_j \cdot x_{ejsd} \quad \forall e \in \mathcal{E}, d \in \mathcal{D} \tag{7.4.15}$$

$$\sum_{d \in \mathcal{D}} z_{ed} \leq h_e^m + u_e \quad \forall e \in \mathcal{E} \tag{7.4.16}$$

$$u_e \leq h_e^m \cdot 0.25 \quad \forall e \in \mathcal{E} \tag{7.4.17}$$

$$\sum_{j \in \mathcal{J}} v_j \cdot b_j \geq B^{min} \tag{7.4.18}$$

$$\sum_{d \in \mathcal{D}^j} \varphi_{jd} = v_j \quad \forall j \in \mathcal{J} \tag{7.4.19}$$

$$\sum_{e \in \mathcal{E}} \sum_{s \in \mathcal{S}} x_{ejsd} \leq \varphi_{jd} \cdot M \quad \forall j \in \mathcal{J}, d \in \mathcal{D}^j \tag{7.4.20}$$

The objective function (7.4.1) minimizes costs arising from overtime hours per employee, weekend hours per employee and weekend day, and penalties costs caused by unserved positions for all activities and skills.

The proposed model variables x_{ejsd}, y_{ed} and w_{ed} are highly interrelated. Their relationship is displayed in Constraints (7.4.2) to (7.4.6).

The required number of employees for each skill in each activity needs to be fulfilled (7.4.7) either through internal or external staff, where the latter is causing a penalty p_{js} in the objective. However, employee e can only be assigned to activity j, if the assignment is allowed by (7.4.8). This constraints also ensure that each employee is only assigned to a maximum of one skill per activity. Constraints (7.4.9) guarantee, that employee e has the required skills for working in job j and skill s. Two activities j on day d and j' on day d' can only be assigned to the same worker, if this is allowed by $f_{jdj'd'}$ in Constraints (7.4.11).

The possible working day patterns for all employees have already been proposed in the above problem description. Each employee needs to have two consecutive days off per week, starting from Mondays, see Constraints (7.4.12). Furthermore, all employees who do not have a weekend contract (summarized in set $\mathcal{E}'$), need to have at least one weekend off per span of four weeks, as in Constraints (7.4.13). The maximum number of consecutive working days is F_{max} (Constraints (7.4.14)). These last constraints are necessary because Constraints (7.4.12) would not prohibit an employee from working from Wednesday in week 1 to Friday in week 2, which would be ten consecutive working days. However, in our case $F_{max} = 9$.

We then calculate the daily working time z_{ed} for each employee e and day d using Constraints (7.4.15) as an intermediate step. We consider overtime per month (where a month consists of four consecutive weeks in our case) using variable u_e

for each employee e (see Constraints (7.4.16)). Overtime is limited to 25% of the regular working hours (Constraints (7.4.17)).

One important constraint in our model is to collect a minimum reward or number of blood donations from the selected activities. This is achieved by Constraint (7.4.18). Then, Constraints (7.4.19) guarantee that only one day d for activity j is chosen from the set of eligible days, $\mathcal{D}^j$. Finally, Constraints (7.4.20) make sure that assignments x_{ejsd} are only done for chosen days φ_{jd}.

Chapter 8

Solution Approach and Computational Results

8.1 Solution Approach

Preliminary experiments have shown that solving the monolithic MIP presented in Section 7.4 in short running time is intractable. Therefore, this section is dedicated to a hierarchical solution method to address the problem formulated in Section 7.4. Section 8.1.1 introduces the problem decomposition and gives some insight into solution evaluation strategies. In Sections 8.1.2 and 8.1.3, we propose two different algorithm variants to address the integrated activity selection and staff scheduling problem.

8.1.1 General Problem Decomposition

We adopt a two-level hierarchical decomposition approach, involving a high-level master problem and a low-level subproblem, as visualized in Figure 8.1. The master problem consists of the job-day selection, i.e., we choose activities which should take place and select a day among the set of eligible days $\mathcal{D}^j$ for each chosen activity in the master problem. We denote the set of chosen jobs in the master problem to $\mathcal{J}^M$ and the corresponding chosen day for job j is abbreviated to d_j^M. From the mixed integer formulation, this would reflect the decisions for variables v_j and φ_{jd}. Additionally, we fix for all employees $e \in \mathcal{E}$ the pattern of *potential* working days and days off duty, respecting Constraints (7.4.12) to (7.4.14) as well as the employee specific availability weekdays. Hence, for each employee e, the master solution contains a binary vector of size D, where an entry 1 at the d-th position indicates that the employee is available to be assigned to activities on day d.

The actual employee-to-activity assignment is done in the subproblem, where

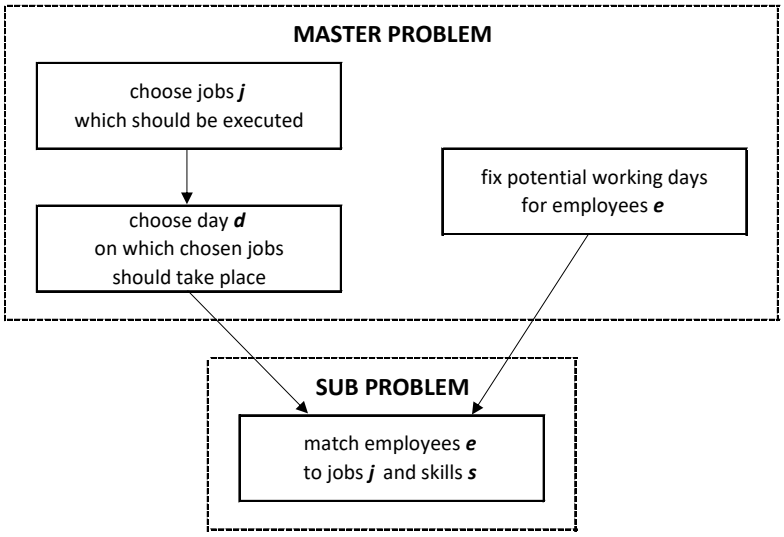


Figure 8.1: Problem Decomposition – Overview.

the final costs of the assignment are revealed. The subproblem formulation can be found in Section 8.1.2.5. Note, that the master problem assigns fixed days off and *potential* working days to each employee e. It is not before the subproblem is solved that the actual working days of the employees are revealed, which form a subset of the potential working days from the master problem.

The objective function consists of three cost types: overtime costs, weekend costs and penalty costs, as depicted in Figure 8.2. While the exact costs are finally revealed when solving the subproblem, we will estimate a staffing cost (weekend and overtime) lower bound based on a relaxation of the master problem already when constructing the master problem, as solving the subproblem exactly is time-consuming (see 8.1.2.4). In addition, we later propose a method to assess penalty costs by solving a modified version of the subproblem that simply optimizes on penalties and neglects other staffing costs, which we call the "penalty MIP".

8.1.2 Algorithm Variant 1: Lower Bound Based Search plus Repair

This section is dedicated to algorithm variant 1 (abbreviated to A1). Algorithm 9 gives an overview of the respective principles we follow. Here, s corresponds to a master solution, while with *currentobj* and *bestobj* we abbreviate the objective function of the incumbent solution s and the best found solution, respectively.

The method begins with an initial master solution, which is improved immediately by reducing master level objective function penalties. Those penalties arise if on a given day d the number of required employees per profession p exceeds the

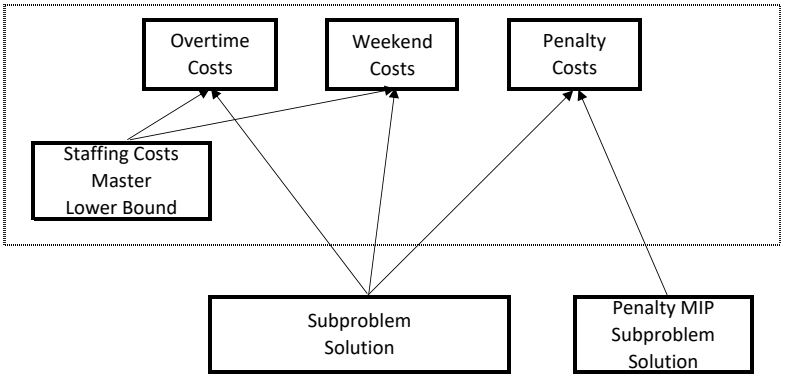


Figure 8.2: Overview of Objective Parts, Corresponding Exact Evaluation and Lower Bound Estimation Approaches.

number of potentially available employees. Afterward, the corresponding subproblem to the initial master problem is solved and the corresponding objective value is calculated, serving as a starting point for the further search. Then, the algorithm repeatedly strives to reduce master level staffing costs through an estimation based variant of a variable neighborhood descent (VND) algorithm proposed by Hansen et al. [2010, 2008] to find better master solutions. Here, the actual objective value of the subproblem is approximated using a lower bound to the true objective value, where potential penalties that might be revealed when solving the subproblem are neglected. The subproblem at the VND local minimum is then solved and evaluated using the procedure SOLVESUBPROBLEMMIP(s), revealing the real overtime and weekend costs as well as penalties arising from unserved positions. In case of penalties, we try to repair the current master problem by replacing or modifying jobs causing penalties, followed by a re-evaluation of the corresponding subproblem. Anytime the incumbent master solution revealed a lower objective than the best known master problem (i.e., $currentobj < bestobj$), the best found objective is replaced by the current. Finally, after the current iteration is finished, we permute the incumbent solution and repeat the search procedure.

The methods used in the pseudo codes are described in more detail in the upcoming Sections 8.1.2.1 to 8.1.2.7.

8.1.2.1 BuildMasterSolution()

Initially, we create a greedy randomized solution to the master problem using simple construction rules. We apply different strategies to the parts of the master problem representation, namely the job-day selection and the initial potential employee schedules.

The initial potential working days and days off for each employee e are assigned

Algorithm 9: A1 – Estimation Based VND plus Repair

```
1  s ← buildMasterSolution();
2  s ← reduceMasterLevelPenalties(s);
3  bestobj ← solveSubProblemMIP(s);
4  repeat
5  |  s ← reduceMasterLevelStaffingCosts(s);
6  |  currentobj ← solveSubProblemMIP(s);
7  |  try ← 0;
8  |  while currentobj.penalty > 0 ∧ try < 10 do
9  |  |  s ← repairPenaltiesFromSubProblemMIP(s);
10 |  |  currentobj ← solveSubProblemMIP(s);
11 |  |  try ← try + 1;
12 |  end
13 |  if currentobj < bestobj then
14 |  |  bestobj ← currentobj;
15 |  end
16 |  s ← permute(s);
17 until termination criterion met;
```

randomly, according to the following strategy: Employees with weekend contracts ($e \notin \mathcal{E}'$) potentially work on any of their available days as all relevant constraints adhere automatically for those employees. The potential schedule of all other employees $e \in \mathcal{E}'$ is generated in a two-stage procedure:

1. First, for every week in the planning horizon, we select two consecutive days and mark them as days off duty. If none of the hereby chosen day-pairs is a complete weekend, we select a random week within the planning horizon and replace the currently chosen non-working day pair of the given week by the corresponding weekend. Hence, we guarantee to respect Constraints (7.4.12) and (7.4.13). The thereby chosen pattern would, however, violate Constraints (7.4.14), for example, if an employee has two consecutive working days off on Monday and Tuesday in the first week and on Saturday and Sunday in the second week, resulting in overall 10 consecutive potential working days. If this is the case, a correction mechanism is applied that iteratively updates the day-pair off for the conflicting weeks until the conflict is resolved.

2. Then, all remaining, yet undecided days, are fixed according to the weekday availability of the employee. For example, if an employee is available on all weekdays but Monday and we fixed days off duty in the given week to be Saturday and Sunday, the employee schedule of the current week corresponds

Tuesday to Friday.

The initial selected job-day combinations are determined as follows: First, we calculate for all jobs the required employee hours, i.e., $\sum_s r_{js} \cdot t_j$. We sort all jobs according to the fraction of demanded hours per unit of reward $\sum_s r_{js} \cdot t_j / b_j$ in non-decreasing order. We then choose the activities from the head of the sorted list of jobs to take place until the minimum reward is reached. The corresponding days on which the selected activities will take place are then chosen according to the following strategy: If either all possible days $\mathcal{D}^j$ are non-weekend days or all eligible days are weekend days, we choose the day purely randomly. If the set of possible days consists of at least one weekend and non-weekend day, then we choose the day randomly among the subset of non-weekend days to avoid costly weekend hours for employees.

8.1.2.2 ReduceMasterLevelPenalties(s)

Master level penalties arise if on a specific day d there are fewer employees of a profession p *potentially* available than are required to fulfill the chosen jobs on the same day. We estimate these penalties in the master problem by comparing the number of required employees per profession to the number of available employees per profession on a given day d. The required employees per profession per day can be calculated by $R_p^d = \sum_{j \in \mathcal{J}_d^M} \sum_{s \in \mathcal{S}^p} r_{js}$ with $\mathcal{J}_d^M$ the set of chosen jobs on day d from the master problem M. The total number of employees on duty per day d is assumed to be E_p^d. If the number of demanded employees per profession on a given day d is larger than the number of employees on duty on the same day, a penalty cost of Pen occurs. Then, $C_{LB}^{pen} = \sum_{p \in \mathcal{P}} \sum_{d \in \mathcal{D}} C_{pd}^{pen}$ forms a lower bound to the penalties of the overall objective, where

$$C_{pd}^{pen} = Pen \cdot \max(R_p^d - E_p^d, 0) \tag{8.1.1}$$

Every time we create a new master solution during the optimization procedure, we evaluate its penalty lower bound like this. In case the current solution includes any master level penalties, we try to repair the solution using a variable neighborhood descent variant, where the objective function to be minimized is formed by C_{LB}^{pen}. The VND then either stops at a local minimum with $C_{LB}^{pen} > 0$, or ideally finds a solution with $C_{LB}^{pen} = 0$. This strategy drastically reduces the number of master solutions to be evaluated by a priori removing solutions which would result in high overall objectives due to high penalties. Algorithm 10 summarizes this VND procedure. The calculation of C_{LB}^{pen} for a master solution s is abbreviated to calculateMasterLevelPenalties(s).

Algorithm 10: ReduceMasterLevelPenalties(s)

```
k ← 1 ;
s* ← s;
penalty* ← calculateMasterLevelPenalties(s);
repeat
    n ← 0;
    while n < N do
        s' ← N_k(s);
        penalty ← calculateMasterLevelPenalties(s');
        if penalty < penalty* then
            penalty* ← penalty;
            s* ← s';
            n ← 0;
            k ← 1;
        end
    end
    k ← k + 1;
until k = k_max;
return s*
```

The VND procedure searches three different neighborhoods for improvements to the incumbent solution s, each of which operates on a different part of the master solution representation. A neighbor in the k-th neighborhood is denoted by $\mathcal{N}_k(s)$. The three neighborhoods are:

1. SwitchRandomJob(s): Randomly select a chosen job j (i.e., $j \in \mathcal{J}^M$) and replace this job by various yet unchosen, random jobs j' (i.e., currently $j' \notin \mathcal{J}^M$), such that the total reward remains above the given minimum reward B^{min}.

2. SwitchDayOfRandomJob(s): Randomly select a chosen job $j \in \mathcal{J}^M$ with multiple possible day options (i.e., $|\mathcal{D}^j| > 1$) and replace the currently chosen day by a random entry from set $\mathcal{D}^j$.

3. RebuildEmployeeSchedule(s): Randomly select an employee e and rebuild the corresponding activity pattern (*potential* days on duty, fixed days off) for this employee, considering his/her availabilities and day-off constraints. When rebuilding the employee activity pattern, we follow the same strategy as when we initially built an employee schedule, see Section 8.1.2.1.

k	Neighborhood Type $\mathcal{N}_k(s)$
1	SwitchRandomJob(s)
2	SwitchDayOfRandomJob(s)
3	RebuildEmployeeSchedule(s)

Table 8.1: Neighborhoods k searched within the VND.

Table 8.1 summarizes those neighborhoods. The search starts with the first neighborhood applying the first improvement strategy, which has shown to be more beneficial than the best improvement strategy in preliminary tests. Due to the high complexity and vast neighborhood size, we impose a limit on the number of evaluated neighbors in each iteration of the VND, abbreviated to N. Here, we fix $N = 500$. If no better solution has been found among N randomly selected neighbors in a given neighborhood, we continue the search in the next neighborhood ($k \leftarrow k + 1$, line 16 in Alg. 10). Every time a better solution is found, the VND continues the search in the first neighborhood ($k \leftarrow 1$, line 13). A local minimum is reached, when none of the neighborhoods $k \in \{1, 2, 3\}$ is able to yield an improving solution. This terminates the VND procedure and we return the best found solution during the VND run (s^*).

8.1.2.3 ReduceMasterLevelStaffingCosts(s)

Algorithmic variant A1 contains a local search procedure, named ReduceMasterLevelStaffingCosts(s), for finding "good" master solutions; Algorithm 11 gives the corresponding pseudo code. Here, our primary optimization objective is the lower bound of overtime and weekend costs, which can be calculated using procedure computeStaffingCostLB(s'). Note, that an exact evaluation of the *actual* subproblem staffing costs is time-consuming, thereby we stick to the lower bound at this point. As before in ReduceMasterLevelPenalties(s), we try to find better solutions in the neighborhoods listed in Table 8.1, once more limiting the maximum number of evaluated neighbors per iteration to $N = 500$.

Algorithm 11: ReduceMasterLevelStaffingCosts(s)

```
k ← 1 ;
s* ← s;
bestobj ← s.costs;
repeat
    n ← 0;
    while n < N do
        s' ← N_k(s*);
        s' ← reduceMasterLevelPenalties(s');
        currentobj ← computeStaffingCostLB(s');
        if currentobj < bestobj then
            bestobj ← currentobj;
            s* ← s';
            n ← 0;
            k ← 1;
        end
    end
    k ← k + 1;
until k = k_max;
return s*
```

8.1.2.4 ComputeStaffingCostLB(s')

We calculate a lower bound on the staffing costs (i.e., overtime and weekend costs) by comparing the required employee hours of the current master problem's job-day assignment to the available employee hours per profession. Let p be the profession (i.e., doctors, nurses, and medical assistants) and let $\tilde{\mathcal{E}}_p$ and $\hat{\mathcal{E}}_p$ be the set of employees of profession p who have a high and standard contract, respectively. Additionally, let $\mathcal{S}_p$ bet the set of skills of profession p and let $\mathcal{J}^M$ be the set of chosen jobs from the master problem.

Overtime Costs The minimum number of required total employee hours can be calculated by summing up all required employee-hours of the chosen jobs, i.e., $H_p^{tot} = \sum_{j \in \mathcal{J}^M} \sum_{s \in \mathcal{S}_p} r_{js} \cdot t_j$. The supplied hours per profession p belong to different categories: $\tilde{\lambda}_p = \sum_{e \in \tilde{\mathcal{E}}_p} h_e^{max}$ is the number of supplied hours per month of employees with "high" contracts working in regular working time and $\hat{\lambda}_p = \sum_{e \in \hat{\mathcal{E}}_p} h_e^{max}$ is the corresponding number of total supplied regular working hours by employees with a "standard" contract. As all regular working hours do not cause additional costs, the total number of hours at no charge is then $\lambda_p = \tilde{\lambda}_p + \hat{\lambda}_p$. Furthermore,

$\tilde{\lambda}_p^+ = \tilde{\lambda}_p \cdot 0.25$ and $\hat{\lambda}_p^+ = \hat{\lambda} \cdot 0.25$ describe the maximum number of supplied overtime hours for both contract types. Hence, we can calculate a lower bound on overtime costs by comparing the required employee hours with the supplied hours:

$$C_{LB,p}^{ot} = \begin{cases} 0 & \iff H_p^{tot} \leq \lambda_p \\ (H_p^{tot} - \lambda_p) \cdot \hat{c}_p^{ot} & \iff \lambda_p < H_p^{tot} \leq \lambda_p + \hat{\lambda}_p^+ \\ \hat{\lambda}_p^+ \cdot \hat{c}_p^{ot} + (H_p^{tot} & \\ \quad - \lambda_p - \hat{\lambda}_p^+) \cdot \tilde{c}_p^{ot} & \iff \lambda_p + \hat{\lambda}_p^+ < H_p^{tot} \leq \lambda_p + \hat{\lambda}_p^+ + \tilde{\lambda}_p^+ \\ \hat{\lambda}_p^+ \cdot \hat{c}_p^{ot} + \tilde{\lambda}_p^+ \cdot \tilde{c}_p^{ot} & \iff H_p^{tot} \geq \lambda_p + \hat{\lambda}_p^+ + \tilde{\lambda}_p^+. \end{cases} \tag{8.1.2}$$

Weekend Costs The total number of demanded employees per profession p on a weekend day $d \in \mathcal{D}^{we}$ can be calculated by summing up all required employees of jobs chosen on weekend day d: $R_p^d = \sum_{j \in \mathcal{J}_d^M} \sum_{s \in \mathcal{S}_p} r_{js}$ where $\mathcal{J}_d^M$ denotes the set of chosen jobs on day d from the master problem. Let $\hat{E}_p^d$ and $\tilde{E}_p^d$ be the number of employees belonging to profession p who are available on day $d \in \mathcal{D}^{we}$ with "standard" and "high" contracts, respectively. Then, $C_{LB,p}^{we} = \sum_{d \in \mathcal{D}^{we}} C_{pd}^{we}$ and

$$C_{pd}^{we} = \begin{cases} 0 & \iff R_p^d \leq \hat{E}_p^d \\ \sum_{i=1}^{R_p^d - \hat{E}_p^d} g_{pd}^i \cdot \tilde{c}_p^{we} & \iff \hat{E}_p^d < R_p^d \leq \hat{E}_p^d + \tilde{E}_p^d \end{cases} \tag{8.1.3}$$

with g_{pd} a non-decreasingly sorted vector containing for each profession p and for all jobs j taking place on a weekend day $d \in \mathcal{D}^{we}$ and for each required skill s r_{js} entries of the job duration, t_j. Thereby we make sure that only the minimum number of weekend hours is assigned to employees with "high" contracts.

8.1.2.5 SolveSubProblemMIP(s)

The subproblem can be solved using commercial MIP solvers such as CPLEX. This section is dedicated to the mathematical formulation of the subproblem. Here, we slightly change the notation from Tables 7.2 to 7.4: a_{ej} is a binary variable, simply indicating whether employee e can work in activity j. In addition to general weekday availabilities and job durations, this input parameter also contains the master decision of an employee being available on a given day d combined with the knowledge of day d_j^M job j will take place. $f_{jj'}$ is a binary input indicating whether two activities j and j' can be served by the same employee, considering minimum rest time and activities that take place on the same day. x_{ejs} corresponds to the reduced version of variable x_{ejsd} from the full model and describes whether

employee e is working in activity j and skill s (index d is no longer needed in the subproblem).

$$\text{minimize} \sum_{e \in \mathcal{E}} u_e \cdot c_e^{ot} + \sum_{e \in \mathcal{E}} \sum_{d \in \mathcal{D}^{we}} z_{ed} \cdot c_e^{we} + \sum_{j \in \mathcal{J}} \sum_{s \in \mathcal{S}} p_{js} \cdot Pen \tag{8.1.4}$$

Subject to:

$$(\sum_{e} x_{ejs}) + p_{js} \geq r_{js} \quad \forall j \in \mathcal{J}, s \in \mathcal{S} \tag{8.1.5}$$

$$\sum_{s} x_{ejs} \leq a_{ej} \quad \forall e \in \mathcal{E}, j \in \mathcal{J}, d \in \mathcal{D}^j \tag{8.1.6}$$

$$x_{ejs} \leq c_{es} \quad \forall e \in \mathcal{E}, j \in \mathcal{J}, s \in \mathcal{S} \tag{8.1.7}$$

$$\sum_{s} x_{ejs} + \sum_{s} x_{ej's} \leq f_{jj'} + 1 \quad \forall e \in \mathcal{E}, j, j' \in \mathcal{J}, j' \neq j \tag{8.1.8}$$

$$z_{ed} = \sum_{j \in \mathcal{J}^d} \sum_{s} t_j \cdot x_{ejs} \quad \forall e \in \mathcal{E} \tag{8.1.9}$$

$$\sum_{j} \sum_{s} t_j \cdot x_{ejs} \leq h_e^m + u_e \quad \forall e \in \mathcal{E} \tag{8.1.10}$$

$$u_e \leq h_e^m \cdot 0.25 \quad \forall e \in \mathcal{E} \tag{8.1.11}$$

The objective function (8.1.4) remains unchanged to the full MIP (7.4.1). Constraints (8.1.5) make sure enough employees with the required skills are assigned to each activity, where unfilled positions are assigned to p_{js} (penalty). Constraints (8.1.6) ensure that employees are only assigned to possible activities. Constraints (8.1.7) assign employees to work in their acquired skills only. In Constraints (8.1.8) we check whether two activities j and j' can be supplied by the same employee. Constraints (8.1.9) calculate the daily working time per employee per day. This figure is later used in Constraints (8.1.10), where the required overtime per employee is calculated. Constraints (8.1.11) limit the overtime per employee to 25% of his regular working time.

8.1.2.6 RepairRenaltiesFromSubProblemMIP(s)

During our search we have put effort in minimizing master level penalties. Nevertheless, in evaluating the master solution exactly by solving the subproblem, multiple constraints might lead to additional penalties through not filled positions:

- The master level assumes unlimited overtime of employees, as it simply compares the *potentially* working employees to the required employees when evaluating master level penalties. In the subproblem, however, overtime is limited to 25% of the regular working time in Constraints (8.1.11).

- The master level assumes each member of a profession p has all relevant skills $s \in \mathcal{S}^p$. The subproblem assignment might, however, reveal missing skills on some days.
- The master level compares available employees to required employees for each day d of the planning horizon separately. Discrepancies between jobs on consecutive days as in Constraints (8.1.8) are neglected.
- The master level does not consider minimum/maximum working time regulations per day for each employee e, as is considered in Constraints (8.1.6).

In case solving the subproblem has revealed penalties, we strive to repair them by slightly modifying the master problem directly at the positions that induce the penalties. We identify from the subproblem solution all jobs j with unserved positions, i.e., jobs with $p_{js} > 0$. If the concerned job might take place on multiple days, i.e., $|\mathcal{D}^j| > 1$, we replace the current day d by a random day from $\mathcal{D}^j$. Otherwise, if the job cannot take place on multiple days, we replace the current job j by various yet unchosen, random jobs j' (i.e., currently $j' \notin \mathcal{J}^M$), such that the total reward remains above the given minimum reward. Note, that the number of attempted repairs is limited to 10.

8.1.2.7 Permute(s)

After each iteration of the algorithm, a permutation strategy needs to be applied such that the next iteration starts the search from a modified master solution. We intensively tested various perturbation variants, namely greedy (G), random (R), and shaking (S). The first two variants correspond to a multi-start local search (Martí et al. [2018]), whereas permutation of the latter type is usually applied in Iterated Local Searches (ILS, Stützle and Ruiz [2018]).

- Greedy (G): The new starting solution of the next iteration is built using the method BuildMasterSolution().
- Random (R): The job-day assignment of the starting solution for the next iteration is selected completely randomly. The employee working patterns are chosen as in BuildMasterSolution().
- Shaking (S): The currently best found master solution undergoes a larger modification, which means it is perturbed by applying methods SwitchRandomJob(s) and SwitchDayOfRandomJob(s) known from neighborhood search various times. The number of perturbation steps is an input parameter to the optimization. Preliminary tests have shown, that perturbing the best found solution 10 times gives good overall results.

Algorithm 12: A2 – Penalty MIP and Lower Bound Based Search

```
1  s ← buildMasterSolution();
2  s ← reduceMasterLevelPenalties(s);
3  bestobj ← solveSubProblemMIP(s);
4  repeat
5  |  s ← penaltyMIPBasedVND(s);
6  |  currentobj ← solveSubProblemMIP(s);
7  |  if currentobj < bestobj then
8  |  |  bestobj ← currentobj;
9  |  end
10 |  s ← permute(s);
11 until termination criterion met;
```

8.1.3 Algorithm Variant 2: Penalty MIP and Lower Bound Based Search

As an alternative approach to algorithm variant 1 (A1), we propose a second variant, abbreviated to A2, which differs from the previous approach in the way the VND procedure looks for better solutions. Algorithm 12 gives an overview of this second method; the main difference to Algorithm 9 is in line 5:

Once more, overtime and weekend costs are estimated based on lower bounds. The penalties, however, are revealed by solving a simplified version of the subproblem, where the objective is minimizing the number of unserved positions and hence the penalty costs instead of the overall costs of overtime, weekend and penalty costs. We call this simplified version of the subproblem the "penalty MIP". Solving the penalty MIP is considerably faster than solving the whole subproblem, hence, we keep on estimating overtime and weekend costs by their lower bounds.

8.1.3.1 PenaltyMIPBasedVND(s)

As in method ReduceMasterLevelStaffingCosts(s), we once more apply a classical VND procedure to find better solutions in algorithmic variant 2 through procedure PenaltyMIPBasedVND(s). The overtime and weekend costs are estimated by their lower bound as was done in the first variant. This version of the algorithm, however, rather calculates penalties arising when solving the subproblem exactly including the reduced objective function:

$$\text{minimize} \sum_{j \in \mathcal{J}^M} \sum_{s \in \mathcal{S}} p_{js} \cdot Pen \tag{8.1.12}$$

Algorithm 13: PenaltyMIPBasedVND(s)

```
1  k ← 1 ;
2  s* ← s;
3  bestobj ← s.costs;
4  repeat
5  |  n ← 0;
6  |  while n < N do
7  |  |  s' ← N_k(s*);
8  |  |  s' ← reduceMasterLevelPenalties(s');
9  |  |  currentobj ← computeStaffingCostLB(s') +
   |  |    solvePenaltyMIP(s');
10 |  |  if currentobj < bestobj then
11 |  |  |  bestobj ← currentobj;
12 |  |  |  s* ← s';
13 |  |  |  n ← 0;
14 |  |  |  k ← 1;
15 |  |  end
16 |  end
17 |  k ← k + 1;
18 until k = k_max;
19 return s*
```

This reduced subproblem is solved every time we evaluate the quality of a neighboring solution in the VND. Due to the fact that solving this "penalty MIP" is time-consuming the number of evaluated neighbors N per iteration in PenaltyMIPBasedVND(s) is drastically reduced to only 10 neighbors. Note, that solving the reduced objective MIP is of course much less time-consuming than solving the subproblem including costs of overtime and weekend hours. Thus, we continue estimating the latter cost aspects by their lower bounds.

Profession	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Doctors	9	10	11	11	13	14	14
Nurses	20	20	20	25	30	30	30
Med. Ass.	57	57	57	57	57	56	56

Table 8.2: Availability of employees during a week from Monday to Sunday.

8.2 Computational Results

The described solution approaches have been implemented and tested with a set of real-world inspired random instances of various sizes and difficulties. We begin with a brief description of the instances and environment used for the computational study (Section 8.2.1). Then, we discuss results of those computational experiments in Section 8.2.2 and present some statistical tests in Section 8.2.3.

8.2.1 Experimental Set-up

The instances were generated using real-world employee and activity characteristics available from the Red Cross Blood Donation Services in Vienna/Austria.

The set of available employees is fixed and consists in total of 101 employees. This set is divided into three professions, namely doctors (14 employees), nurses (31) and medical assistants (56). Three doctors and ten nurses have a special weekend contract, meaning that they are only available from Thursday or Friday to Sunday and the constraint of a minimum of one weekend off per four week time period is not required for those employees. On average, doctors have 24.6 contracted working hours per week and are available on 5.9 days per week; nurses have 25.5 contracted working hours per week and are available on 5.8 days per week and medical assistants have 30.6 contracted working hours per week and are available on 7.0 days per week on average. Doctors possess on average 1.9 skills, nurses have on average 1.8 skills and medical assistants on average have 2.3 skills out of a total set of seven different skills. 79% of doctors have a standard contract; within the group of nurses, 63% have a standard contract and finally, among medical assistants, only 35% have a standard contract. Employees with a 40-hour contract per week additionally can only work in jobs with minimum and maximum duration of 7 and 9 hours, respectively. For all other employees, there is no minimum or maximum daily active time.

The availability of the three professions on the weekdays from Monday to Sunday is summarized in Table 8.2. As can be seen, the number of available doctors and nurses increases during the week and is highest on the weekend, while the number of available medical assistants slightly decreases during the weekend.

The jobs per instance have been drawn randomly from a set of total 1453 activities. The average activity duration within this set was 8.2 hours (min: 2.8, max: 12.3 hours), which includes the actual activity time as well as the travel time. The average reward per activity is 94.5 (min: 15, max: 350); the average reward per activity hour is 11.4 (min: 2.7, max: 37.7). On average, an activity requires 8 employees, 66 employee-hours or 0.7 employee-hours per unit of reward (min: 0.3, max: 1.8 employee-hours/reward unit). The available days per activity are assigned randomly to the activities, where we assume that 1/3 of activities are pure weekend activities, so they can only take place on Saturdays and Sundays. The average number of available days per activity varies between 2 and 5, depending on the instance size and complexity.

The costs per profession and contract type are listed in Table 7.1. Additionally, we consider penalty costs during the optimization phase, where each unassigned position causes penalties of $Pen = 1 = 000$, no matter which skill is unassigned. As we strive to minimize costs, finding solutions without unassigned positions is of utmost importance.

All algorithms have been implemented in C++. The MIP was solved using CPLEX 12.7. The experiments have been carried out on the Vienna Scientific Cluster (VSC3), whose compute nodes are equipped with two Intel Xeon E5-2650v2, 2.6 GHz, 8 core CPUs each.

8.2.2 Results of Computational Experiments

Tables 8.3 to 8.7 summarize the results of computational tests performed on 28 real-world inspired problem instances. The instances are described by three numbers. The first number indicates the number of eligible activities J. The second figure depicts the average number of possible days for activities j, $|\mathcal{D}^j|$. Finally, the third number describes the workload as the minimum rewards needed (B^{min}). For example, 150-3-11500 denotes an instance including 150 activities, which on average can take place on three eligible days, and the minimum required blood donations (reward) is 11500. Columns "Comp." specify the complexity of the instance, as the product of jobs to choose from times the average number of days per job. Columns "Bound" give the best-known lower bound for each instance. Note, that we split the 28 instances into two groups: (1) instances with best lower bound of 0 and (2) instances with best lower bound larger than zero. We sort the instances within those groups according to their complexity.

We compare the two proposed algorithmic variants – lower bound based search plus repair (A1) and the penalty MIP and lower bound based search (A2) – to a CPLEX optimization run. For each algorithmic variant, we test multiple restart strategies, namely a greedy restart (G) a random restart (R) and a perturbation-based restart (P), as described in Section 8.1.2.7. We conducted tests considering

time limits of 10, 60, 120 and 600 minutes.

The results show the superiority of all algorithmic variants against the commercial solver, especially for small running times. For example, five of the six algorithmic variants have found the optimal solution with objective function 0 for instance 200-3-12000 (third line in Table 8.3) within 10 minutes, while the exact approach achieved a result of 153000 given the same time limit. Another example is instance 150-3-12500, for which algorithmic variants A1 find solutions close to the lower bound within 10 minutes, while CPLEX obtains an integer solution with an objective value of 120348 given the same running time. After 60 and 120 minutes (Tables 8.4 and 8.5), the commercial solver catches up only in a few instances but still fails to find reasonable results for most challenging instances with higher complexity. Finally, we report the results of an extensive running time of 600 minutes in Table 8.6. Here, the exact approach using CPLEX seems to outmatch the proposed methods in many instances. Nevertheless, there exist multiple instances for which the exact solver could not outperform the heuristic approach after 10 hours of running time. Note, that for none of the instances in the second group (i.e., those with bound larger than 0), CPLEX could achieve a proven to be optimal solution. When comparing algorithm variants A1 and A2, we see that the first algorithm outperforms the second approach on average. We conclude, that investing in a more detailed evaluation of penalties within the subproblem is not beneficial as it is too time-consuming compared to the pure estimation approach which repairs penalties in case they are observed.

The further analysis focuses on algorithmic variant "A1G", as this method seems to be the most promising among the tested variants. Table 8.7 directly compares the proposed variant with CPLEX given the mentioned time limits. The bold numbers indicate the best found solution up until the current time limit. After 10 minutes of running time, A1G already found reasonable results for all instances, while the commercial solver seems to have trouble finding solutions not containing penalties for unserved positions (reflected by the high objective values). This picture only changes slightly for problems with less complexity after running times of 60 and 120 minutes. It is worth mentioning that without statistical testing, we cannot assess which method works better for extensive running times of 10 hours. However, from Table 8.7 we can see, that in those cases, where CPLEX is better than A1G, the difference among the objective values is considerably smaller than for the opposite cases where A1G performs better than the commercial solver.

Figure 8.3 gives further insight into the algorithmic performance of the proposed heuristic A1G against CPLEX over time. Subfigure (a) contains a solution with lower bound 0. This best solution was found by the heuristic method within a few seconds, while CPLEX reaches this solution only after 470 minutes. The same lower bound of 0 is visible also in Subfigure (b). However, this bound can be obtained by

Instance	Comp.	Bound	CPLEX	A1R	A1G	A1S	A2R	A2G	A2S
150-3-11500	450	0	1062	**0**	**0**	4	12	7	916
150-3-12000	450	0	123000	85	**79**	90	124	128	145
200-3-12000	600	0	153000	**0**	**0**	**0**	**0**	**0**	2115
200-3-12500	600	0	146085	**0**	**0**	**0**	**0**	**0**	2570
200-3-13000	600	0	149000	**0**	**0**	**0**	2	**0**	2261
200-3-13500	600	0	2879	36	**23**	50	117	88	70
300-2-15000	600	0	137007	**116**	131	131	756	642	1965
150-5-12000	750	0	1247720	**0**	**0**	2	4	**0**	2577
150-5-12500	750	0	30228	15	**7**	35	57	41	1521
250-3-13000	750	0	1147	**0**	**0**	**0**	**0**	**0**	1228
250-3-13500	750	0	14261	**0**	**0**	1	3	**0**	2076
250-3-14000	750	0	11116	29	**10**	28	136	36	47
250-3-14500	750	0	1978720	89	**84**	96	659	512	877
150-3-12500	450	343	120348	390	391	**387**	439	435	765
150-3-13000	450	735	125697	817	808	**803**	838	838	1762
200-3-14000	600	42	1573720	334	**300**	306	751	644	646
200-3-14500	600	791	153117	1099	1130	**1053**	1550	1541	2366
200-3-15000	600	1753	176055	2201	2221	**2152**	2580	2610	3318
300-2-15500	600	215	121035	**665**	697	715	1571	1228	2509
300-2-16000	600	805	20941	1355	**1314**	1338	2549	2305	5774
300-2-16500	600	1458	134321	**2169**	2177	2171	10326	5850	8817
300-2-17000	600	2353	2416720	**3768**	3809	4198	14908	14909	14489
150-5-13000	750	101	1247720	**186**	190	202	291	285	1219
150-5-13500	750	659	1247720	828	806	**803**	986	983	3341
150-5-14000	750	1512	1247720	1690	1678	**1666**	1790	1794	2151
150-5-14500	750	2438	1247720	**2756**	2786	2759	2874	2832	3698
250-3-15000	750	277	1978720	637	**567**	640	1445	1189	1255
250-3-15500	750	958	1978720	**1299**	1334	1336	2233	2078	2592

Table 8.3: Run time 10 minutes: Results of 28 instances with varying complexity. "Bound" gives the best-known lower bound to the problem objective. "CPLEX" gives results of exact optimization using MIP Solver CPLEX. Columns "A1R" to "A2S" give average results of 16 random replications for each proposed method. Bold numbers represent the best solution per line.

Instance	Comp.	Bound	CPLEX	A1R	A1G	A1S	A2R	A2G	A2S
150-3-11500	450	0	**0**	**0**	**0**	**0**	**0**	**0**	916
150-3-12000	450	0	**15**	65	65	76	98	107	120
200-3-12000	600	0	**0**	**0**	**0**	**0**	**0**	**0**	1990
200-3-12500	600	0	3013	**0**	**0**	**0**	**0**	**0**	2570
200-3-13000	600	0	1430	**0**	**0**	**0**	**0**	**0**	2261
200-3-13500	600	0	263	18	**13**	30	38	34	45
300-2-15000	600	0	2365	**73**	79	92	647	501	1822
150-5-12000	750	0	1645	**0**	**0**	**0**	**0**	**0**	2577
150-5-12500	750	0	363	3	**0**	17	22	13	1520
250-3-13000	750	0	155	**0**	**0**	**0**	**0**	**0**	1224
250-3-13500	750	0	489	**0**	**0**	**0**	1	**0**	2076
250-3-14000	750	0	7432	16	**3**	11	41	11	35
250-3-14500	750	0	8198	63	**58**	79	536	389	473
150-3-12500	450	343	**374**	376	378	376	418	414	758
150-3-13000	450	735	863	**778**	786	779	815	814	1157
200-3-14000	600	42	16567	273	**255**	273	627	563	605
200-3-14500	600	791	12921	1008	1004	**983**	1444	1384	1448
200-3-15000	600	1753	18418	2018	2029	**2008**	2440	2434	2764
300-2-15500	600	215	12849	578	**565**	589	1242	1089	1180
300-2-16000	600	805	5271	1203	**1191**	1201	2062	1836	2322
300-2-16500	600	1458	12697	1947	2007	**1921**	3073	2761	5293
300-2-17000	600	2353	64057	3005	2970	**2961**	9283	7325	14071
150-5-13000	750	101	20338	169	**162**	313	243	250	678
150-5-13500	750	659	242377	748	750	**740**	918	913	2319
150-5-14000	750	1512	253664	1641	1633	**1616**	1743	1729	1808
150-5-14500	750	2438	109672	2674	2681	**2644**	2732	2743	3487
250-3-15000	750	277	5820	552	**513**	555	1172	1052	1202
250-3-15500	750	958	45195	1229	**1202**	1244	2005	1866	2012

Table 8.4: Run time 60 minutes: Results of 28 instances with varying complexity. "Bound" gives the best-known lower bound to the problem objective. "CPLEX" gives results of exact optimization using MIP Solver CPLEX. Columns "A1R" to "A2S" give average results of 16 random replications for each proposed method. Bold numbers represent the best solution per line.

Instance	Comp.	Bound	CPLEX	A1R	A1G	A1S	A2R	A2G	A2S
150-3-11500	450	0	**0**	**0**	**0**	**0**	**0**	**0**	916
150-3-12000	450	0	**15**	62	55	69	89	93	114
200-3-12000	600	0	**0**	**0**	**0**	**0**	**0**	**0**	1990
200-3-12500	600	0	**0**	**0**	**0**	**0**	**0**	**0**	2570
200-3-13000	600	0	47	**0**	**0**	**0**	**0**	**0**	2261
200-3-13500	600	0	109	17	**9**	24	32	20	42
300-2-15000	600	0	**0**	67	68	84	593	463	524
150-5-12000	750	0	**0**	**0**	**0**	**0**	**0**	**0**	2577
150-5-12500	750	0	**0**	**0**	**0**	14	15	7	1520
250-3-13000	750	0	**0**	**0**	**0**	**0**	**0**	**0**	1161
250-3-13500	750	0	96	**0**	**0**	**0**	1	**0**	2076
250-3-14000	750	0	7432	13	**1**	8	36	7	30
250-3-14500	750	0	1382	53	**51**	70	493	375	428
150-3-12500	450	343	**363**	372	374	372	413	409	430
150-3-13000	450	735	836	776	782	**773**	810	813	825
200-3-14000	600	42	1038	256	**248**	256	580	527	586
200-3-14500	600	791	977	993	985	**968**	1394	1358	1421
200-3-15000	600	1753	18418	**1986**	1996	1998	2409	2366	2755
300-2-15500	600	215	2688	**547**	550	559	1212	1052	1126
300-2-16000	600	805	1375	**1171**	1174	1194	1992	1774	1886
300-2-16500	600	1458	2118	1905	1931	**1888**	2960	2734	3263
300-2-17000	600	2353	5618	2913	2891	**2873**	6912	5325	13683
150-5-13000	750	101	14311	162	**161**	163	229	221	411
150-5-13500	750	659	8861	744	745	**722**	904	898	2000
150-5-14000	750	1512	253664	1625	1619	**1602**	1729	1726	1786
150-5-14500	750	2438	109672	2655	2651	**2618**	2707	2730	3285
250-3-15000	750	277	1492	507	**506**	535	1128	1022	1153
250-3-15500	750	958	16433	1209	**1188**	1231	1971	1818	1984

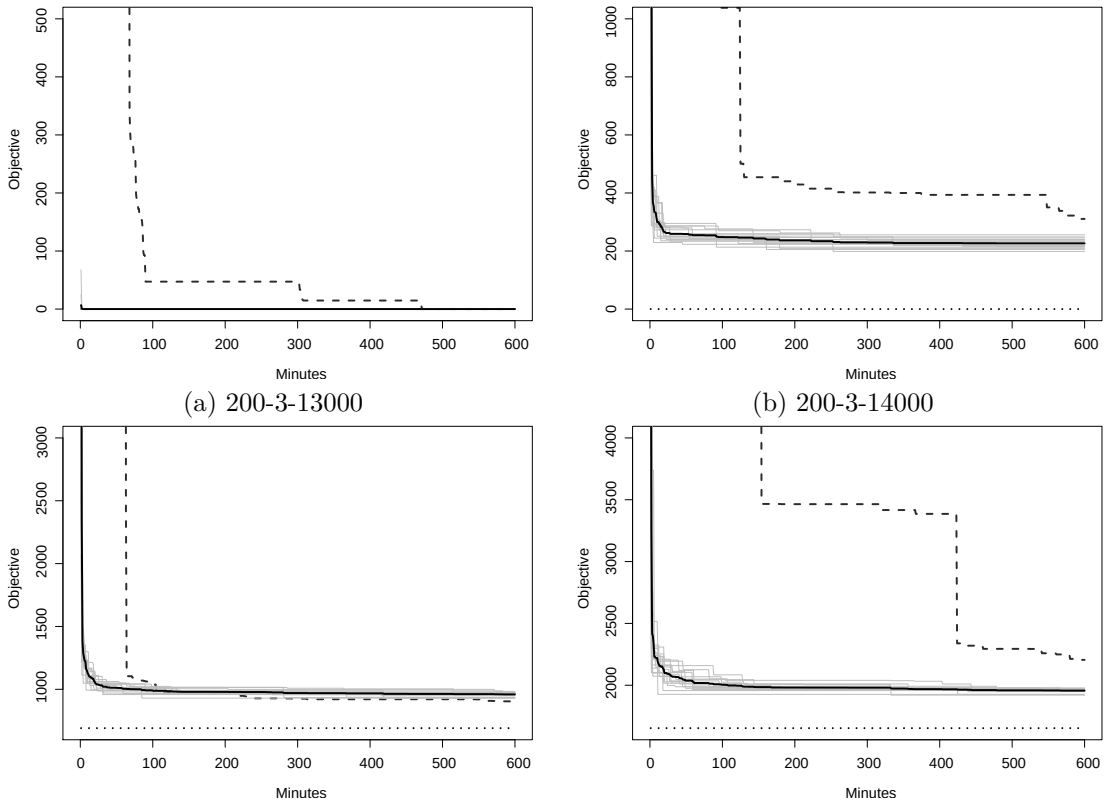
Table 8.5: Run time 2 hours: Results of 28 instances with varying complexity. "Bound" gives the best-known lower bound to the problem objective. "CPLEX" gives results of exact optimization using MIP Solver CPLEX. Columns "A1R" to "A2S" give average results of 16 random replications for each proposed method. Bold numbers represent the best solution per line.

Instance	Comp.	Bound	CPLEX	A1R	A1G	A1S	A2R	A2G	A2S
150-3-11500	450	0	**0**	**0**	**0**	**0**	**0**	**0**	916
150-3-12000	450	0	**15**	48	49	59	76	76	106
200-3-12000	600	0	**0**	**0**	**0**	**0**	**0**	**0**	1600
200-3-12500	600	0	**0**	**0**	**0**	**0**	**0**	**0**	2453
200-3-13000	600	0	**0**	**0**	**0**	**0**	**0**	**0**	2261
200-3-13500	600	0	109	8	**3**	14	21	9	40
300-2-15000	600	0	**0**	43	50	65	495	404	449
150-5-12000	750	0	**0**	**0**	**0**	**0**	**0**	**0**	2577
150-5-12500	750	0	**0**	**0**	**0**	10	1	**0**	1520
250-3-13000	750	0	**0**	**0**	**0**	**0**	**0**	**0**	1161
250-3-13500	750	0	**0**	**0**	**0**	**0**	1	**0**	1945
250-3-14000	750	0	45	5	**0**	4	19	2	28
250-3-14500	750	0	**28**	40	34	55	404	323	390
150-3-12500	450	343	**360**	367	368	367	402	400	411
150-3-13000	450	735	**740**	771	770	767	802	799	813
200-3-14000	600	42	311	**226**	**226**	240	514	470	539
200-3-14500	600	791	**904**	969	959	940	1326	1284	1370
200-3-15000	600	1753	2205	**1951**	1956	1953	2316	2284	2394
300-2-15500	600	215	**462**	507	505	532	1097	997	1064
300-2-16000	600	805	**1046**	1133	1129	1161	1819	1674	1818
300-2-16500	600	1458	1924	1840	**1825**	1839	2806	2584	2638
300-2-17000	600	2353	**2640**	2819	2781	2756	5130	4040	13463
150-5-13000	750	101	267	**146**	149	151	200	191	394
150-5-13500	750	659	**689**	729	720	704	861	860	953
150-5-14000	750	1512	**1519**	1588	1603	1579	1692	1680	1745
150-5-14500	750	2438	**2504**	2607	2605	2583	2659	2683	2721
250-3-15000	750	277	1068	487	**481**	507	1052	948	1036
250-3-15500	750	958	1638	1183	**1161**	1197	1854	1763	1877

Table 8.6: Run time 10 hours: Results of 28 instances with varying complexity. "Bound" gives the best-known lower bound to the problem objective. "CPLEX" gives results of exact optimization using MIP Solver CPLEX. Columns "A1R" to "A2S" give average results of 16 random replications for each proposed method. Bold numbers represent the best solution per line.

			10 minutes		60 minutes		2 hours		10 hours	
Instance	Comp.	Bound	CPLEX	A1G	CPLEX	A1G	CPLEX	A1G	CPLEX	A1G
150-3-11500	450	0	1062	**0**	**0**	**0**	**0**	**0**	**0**	**0**
150-3-12000	450	0	123000	**79**	**15**	65	**15**	55	**15**	49
200-3-12000	600	0	153000	**0**	**0**	**0**	**0**	**0**	**0**	**0**
200-3-12500	600	0	146085	**0**	3013	**0**	**0**	**0**	**0**	**0**
200-3-13000	600	0	149000	**0**	1430	**0**	47	**0**	**0**	**0**
200-3-13500	600	0	2879	**23**	263	**13**	109	**9**	109	**3**
300-2-15000	600	0	137007	**131**	2365	**79**	**0**	68	**0**	50
150-5-12000	750	0	1247720	**0**	1645	**0**	**0**	**0**	**0**	**0**
150-5-12500	750	0	30228	**7**	363	**0**	**0**	**0**	**0**	**0**
250-3-13000	750	0	1147	**0**	155	**0**	**0**	**0**	**0**	**0**
250-3-13500	750	0	14261	**0**	489	**0**	96	**0**	**0**	**0**
250-3-14000	750	0	11116	**10**	7432	**3**	7432	**1**	45	**0**
250-3-14500	750	0	1978720	**84**	8198	**58**	1382	**51**	**28**	34
150-3-12500	450	343	120348	**391**	**374**	378	**363**	374	**360**	368
150-3-13000	450	735	125697	**808**	863	**786**	836	**782**	**740**	770
200-3-14000	600	42	1573720	**300**	16567	**255**	1038	**248**	311	**226**
200-3-14500	600	791	153117	**1130**	12921	**1004**	**977**	985	**904**	959
200-3-15000	600	1753	176055	**2221**	18418	**2029**	18418	**1996**	2205	**1956**
300-2-15500	600	215	121035	**697**	12849	**565**	2688	**550**	**462**	505
300-2-16000	600	805	20941	**1314**	5271	**1191**	1375	**1174**	**1046**	1129
300-2-16500	600	1458	134321	**2177**	12697	**2007**	2118	**1931**	1924	**1825**
300-2-17000	600	2353	2416720	**3809**	64057	**2970**	5618	**2891**	**2640**	2781
150-5-13000	750	101	1247720	**190**	20338	**162**	14311	**161**	267	**149**
150-5-13500	750	659	1247720	**806**	242377	**750**	8861	**745**	**689**	720
150-5-14000	750	1512	1247720	**1678**	253664	**1633**	253664	**1619**	**1519**	1603
150-5-14500	750	2438	1247720	**2786**	109672	**2681**	109672	**2651**	**2504**	2605
250-3-15000	750	277	1978720	**567**	5820	**513**	1492	**506**	1068	**481**
250-3-15500	750	958	1978720	**1334**	45195	**1202**	16433	**1188**	1638	**1161**

Table 8.7: Direct comparison of method A1G to CPLEX for 28 instances after running times of 10 minutes, 60 minutes, 2 hours, and 10 hours. Columns "A1G" give average results of 16 random replications of proposed approach A1G.



(a) 200-3-13000 (b) 200-3-14000

(c) 200-3-14500 (d) 200-3-15000

Figure 8.3: Algorithmic performance of 16 runs of A1G (gray lines) and average among those 16 runs (black line) compared to CPLEX performance (dashed line) and the best-known lower bound (dotted line). Evolution of results for running times between 0 and 600 minutes.

neither the heuristic nor the exact approach, with a clear advantage of the heuristic method which reaches an acceptable objective level quite rapidly. Subfigure (c) shows a slightly different picture: Even though the heuristic method outperforms CPLEX during the first two hours, the commercial solver finds a superior objective value to the average heuristic solution afterward. Subfigure (d) displays one more instance for which the commercial solver is clearly not competitive to the proposed solution approach, neither for shorter nor for longer running times.

8.2.3 Statistical Tests

To assess the quality of the different approaches, we performed statistical tests. One-sample Wilcoxon-Mann-Whitney (WMW) tests have been conducted for pairwise comparing samples including 16 random runs of the heuristic algorithms to the CPLEX results after multiple time steps (i.e., 10, 60, 120 and 600 minutes), see Table 8.8. Two-sample WMW tests were computed to contrast the heuristic approaches, where we focus on the pair-wise comparison of the best found algorithm A1G to the other mentioned approaches, as shown in Table 8.9. The values in the tables indicate the number of instances for which either method revealed significantly better results than the other or there is no statistically significant difference among the results of the compared methods given $\alpha = 0.05$, indicated by "eq.".

Table 8.8 once more confirms the superiority of the six heuristic approaches compared to CPLEX specifically for low running times. For example, in case running time is limited to 10 minutes, all instances can be solved significantly better using algorithmic variants A1G, A1R, A1S, A2G, and A2R. For variant A2S this holds at least for 26 out of the 28 tested instances. Even for larger running times, algorithm variants A1 outperform the commercial solver in 8 instances.

Finally, the statistical tests comparing variant A1G to the other tested approaches strongly supports the theory of this approach being the best among the tested heuristic algorithms. While the tests reveal that A1G and A1R perform equally well on many instances (23 to 25 instances, depending on the running time), A1G approaches between two and five instances significantly better than A1R. The opposite is only true for one single instance. When comparing A1G to the other variants, results are even more considerable in favor of A1G, as shown in Table 8.9.

	A1G vs. CPLEX			A1R vs. CPLEX			A1S vs. CPLEX		
time	A1G	eq.	CPLEX	A1R	eq.	CPLEX	A1S	eq.	CPLEX
10	28	0	0	28	0	0	28	0	0
60	24	2	2	24	3	1	24	3	1
120	18	7	3	18	6	4	18	6	4
600	8	9	11	8	8	12	8	7	13

	A2G vs. CPLEX			A2R vs. CPLEX			A2S vs. CPLEX		
time	A2G	eq.	CPLEX	A2R	eq.	CPLEX	A2S	eq.	CPLEX
10	28	0	0	28	0	0	26	2	0
60	24	2	2	24	2	2	18	4	6
120	15	6	7	15	5	8	12	1	15
600	4	8	16	3	9	16	2	2	24

Table 8.8: Results of one-sample Wilcoxon-Mann-Whitney tests of the heuristic variants A1G, A1R, A1S, A2G, A2R, and A2S versus CPLEX. 28 instances with 16 random replications of the heuristic algorithm are tested against the best found intermediate solution by CPLEX after limited running time. Values indicate the number of instances falling into one of the three categories: (1) Heuristic is significantly better (method abbreviation as column header), (2) no significant difference among the results of the heuristics and CPLEX ("eq."), (3) CPLEX is performing significantly better ("CPLEX").

	A1G vs. A1R			A1G vs. A1S			A1G vs. A2G			A1G vs. A2R			A1G vs. A2S		
time	A1G	eq.	A1R	A1G	eq.	A1S	A1G	eq.	A2G	A1G	eq.	A2R	A1G	eq.	A2S
10	4	24	0	6	21	1	21	7	0	23	5	0	28	0	0
60	2	25	1	7	18	3	21	7	0	21	7	0	28	0	0
120	2	25	1	7	17	4	21	7	0	21	7	0	28	0	0
600	4	23	1	10	14	4	19	9	0	20	8	0	28	0	0

Table 8.9: Results of two-sample Wilcoxon-Mann-Whitney tests of the best performing heuristic variant A1G versus the other proposed variants A1R, A1S, A2G, A2R, and A2S. 28 instances with 16 random replications of each heuristic algorithm are tested. Values indicate the number of instances falling into one of the three categories: (1) A1G is significantly better ("A1G"), (2) no significant difference between A1G and the other method ("eq."), (3) the heuristic method tested against is significantly better (method abbreviation as column header).

Chapter 9

Conclusions to Part II

The integrated activity selection and staff scheduling problem is a highly complex optimization problem that for example arises for blood donation services, that need to select activities to take place from a set of potential activities on various eligible days and then assign skilled employees to the chosen activity-day combinations. In realizing that solving the corresponding MIP for real-world inspired instances is inefficient and time-consuming, we developed a problem decomposition approach, where the master problem is solved using local search techniques that make use of an approximation of the lower bound to the subproblem. In contrast, the subproblem is solved exactly using the MIP solver CPLEX. We summarize, that the proposed algorithmic variants clearly outperform the exact solution approach of the full problem especially for low running times. After a running time of 10 minutes, our heuristic A1G returns for all 28 real-world inspired instances considerably lower costs than the commercial solver. After running time of 10 hours, A1G could still improve on the solution found by CPLEX in 8 instances and returns the same solution in 9 instances. Hence, the proposed algorithm is still competitive even for running times of multiple hours.

Future research on the combination of activity and staff scheduling will be dedicated to accelerated techniques for solving the subproblem, for example by enhancing the model or solving the subproblem using alternative approaches like constraint programming or a fast metaheuristic. Additionally, we strive to solve the full model optimally using exact decomposition approaches like Bender's decomposition.

Chapter 10

Summary and Outlook

This work focuses on solution techniques for rich and real-world scheduling problems in health care. We present two real-world scheduling problems: A patient appointment scheduling problem for patients receiving recurring radiotherapy treatments and an integrated activity selection and staff scheduling problem that arises for example in the case of blood donation services.

We address the radiotherapy patient appointment scheduling problem both deterministically for a long-term planning horizon and stochastically for short-term planning horizons. While the former approach mainly focuses on optimizing the usage of the primary resource – the particle beam – the latter approach also considers the minimization of patient waiting time as an optimization goal. We formulate the problems mathematically, but reason from preliminary tests, that solving the problems exactly is intractable. Hence, we develop (meta)heuristic solution approaches, namely a genetic algorithm and an iterated local search variant to search for good solutions in reasonable running times. We conclude that a combination of the proposed approaches is beneficial in the long-term planning.

For the stochastic, short-term optimization we deploy a proactive-reactive approach. Here, we strive to find baseline schedules which are robust to real-time changes in activity durations. We introduce a buffer parameter as the percentile of the activity duration distribution and find the optimal buffer parameters depending on different objective function weights. Results of intensive computational tests indicate, that adding buffers to the planned activity durations is highly beneficial in terms of minimizing patient waiting time and only slightly increase beam active time (i.e., the second objective). We conclude that stochastic optimization strategies are particularly beneficial in case of low time buffers. For larger buffer sizes, an estimation-based variant of patient waiting time as well as the deterministic variant of the algorithm both show comparable results.

We follow a similar methodological strategy when embracing the second real-world problem. When formulating the problem mathematically and solving it

using a commercial solver, we realize, that the running time to reach solutions of reasonable quality is extensive. Hence, we develop heuristic decomposition approaches which divide the problem into a master problem and a subproblem. The former consists of the selection of activities to take place among a set of eligible activities, the determination of the activity's dates and the *potential* working days of the employees. The subproblem then determines the assignment of employees to activities and skills, given the master level decisions. As solving the subproblem is time-consuming, we strive to find good quality master solutions applying a lower bound estimation of the subproblem. Only in case we reach a good quality master solution using a local search based strategy, we solve the subproblem using an exact solver. Through computational tests on real-world inspired problem instances we conclude, that our optimization algorithms clearly outperform the commercial solver especially for low running times of a few minutes or a few hours. In the case of increased running times, the best among the proposed algorithms still delivers results comparable to those of the exact approach.

Scheduling problems in health care are highly interesting from a researcher's perspective. There exist multiple possibilities for future research on this topic. Specifically, patient waiting time as an objective to scheduling problems has not been addressed widely, while many authors point to the importance of low patient waiting time for their recovery and well-being (e.g., Hall [2012]). The general assumption of patients arriving for their appointments punctually is also questionable. A comparable difficulty arises in case staff members call in sick and have to be replaced by colleagues on duty in staff scheduling problems. Hence, a future research might focus on the uncertainty of scheduling problems in health care, where multiple different strategies can be applied, such as stochastic programming or proactive/reactive scheduling.

Bibliography

M. Affenzeller and S. Wagner. SASEGASA: A new generic parallel evolutionary algorithm for achieving highest quality results. *Journal of Heuristics*, 10(3): 243–267, 2004.

M. Affenzeller, S. Wagner, S. Winkler, and A. Beham. *Genetic Algorithms and Genetic Programming*. CRC Press, Boca Raton, London, New York, 2009. ISBN 978-1584886297.

A. Ahmadi-Javid, Z. Jalali, and K. J. Klassen. Outpatient appointment systems in healthcare: A review of optimization studies. *European Journal of Operational Research*, 258(1):3–34, 2017. ISSN 03772217. doi: 10.1016/j.ejor.2016.06.064.

J. F. Bard and H. W. Purnomo. Preference scheduling for nurses using column generation. *European Journal of Operational Research*, 164(2):510–534, 2005. ISSN 03772217. doi: 10.1016/j.ejor.2003.06.046.

M. A. Begen, R. Levi, and M. Queyranne. Technical note – A sampling-based approach to appointment scheduling. *Operations Research*, 60(3):675–681, 2012. ISSN 0030-364X. doi: 10.1287/opre.1120.1053. URL `http://pubsonline.informs.org/doi/abs/10.1287/opre.1120.1053`.

J. Belien and E. Demeulemeester. Integer programming for building robust surgery schedules. Technical report, Katholieke Universiteit Leuven, Department of Applied Economics, Research Report OR0446, 2004.

B. P. Berg, B. T. Denton, S. Ayca Erdogan, T. Rohleder, and T. Huschka. Optimal booking and scheduling in outpatient procedure centers. *Computers and Operations Research*, 50:24–37, 2014. ISSN 03050548. doi: 10.1016/j.cor.2014.04.007. URL `http://dx.doi.org/10.1016/j.cor.2014.04.007`.

L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8 (2):239–287, 2009. ISSN 15677818. doi: 10.1007/s11047-008-9098-4.

F. Bray, A. Jemal, N. Grey, J. Ferlay, and D. Forman. Global cancer transitions according to the Human Development Index (2008-2030): a population-based study. *The Lancet Oncology*, 13(8):790–801, 2012.

E. K. Burke, P. Leite-Rocha, and S. Petrovic. An integer linear programming model for the radiotherapy treatment scheduling problem. *CoRR*, abs/1103.3, 2011.

B. Cardoen, E. Demeulemeester, and J. Beliën. Operating room planning and scheduling: A literature review. *European Journal of Operational Research*, 201(3):921–932, 2010. ISSN 03772217. doi: 10.1016/j.ejor.2009.04.011. URL `http://dx.doi.org/10.1016/j.ejor.2009.04.011`.

E. Castro and S. Petrovic. Combined mathematical programming and heuristics for a radiotherapy pre-treatment scheduling problem. *Journal of Scheduling*, 15: 333–346, 2012.

T. Cayirli and E. Veral. Outpatient scheduling in health care: a review of literature. *Production and Operations Management*, 12(4):519–549, 2003.

D. Conforti, F. Guerriero, and R. Guido. Optimization models for radiotherapy patient scheduling. *4OR*, 6:263–278, 2008.

D. Conforti, F. Guerriero, and R. Guido. Non-block scheduling with priority for radiotherapy treatments. *European Journal of Operational Research*, 201:289–296, 2010.

D. Conforti, F. Guerriero, R. Guido, and M. Veltri. An optimal decision-making approach for the management of radiotherapy patients. *OR Spectrum*, 33(1): 123–148, 2011. ISSN 0171-6468. doi: 10.1007/s00291-009-0170-y.

S. Dahmen and M. Rekik. Solving multi-activity multi-day shift scheduling problems with a hybrid heuristic. *Journal of Scheduling*, 18:207–223, 2015. doi: 10.1007/s10951-014-0383-z.

G. B. Dantzig and M. N. Thapa. *Linear Programming: Introduction*. Springer, New York, 1997.

M. Davari and E. Demeulemeester. The proactive and reactive resource-constrained project scheduling problem. *Journal of Scheduling*, 2017. ISSN 1099-1425. doi: 10.1007/s10951-017-0553-x. URL `https://doi.org/10.1007/s10951-017-0553-x`. Available online.

P. De Bruecker, J. Van den Bergh, J. Beliën, and E. Demeulemeester. Workforce planning incorporating skills: State of the art. *European Journal of Operational Research*, 243:1–16, 2015. doi: 10.1016/j.ejor.2014.10.038.

P. De Causmaecker and G. Van Den Berghe. A categorisation of nurse rostering problems. *Journal of Scheduling*, 14(1):3–16, 2011. ISSN 10946136. doi: 10.1007/s10951-010-0211-z.

E. Demeulemeester, W. Herroelen, and R. Leus. Proactive-reactive Project Scheduling. In C. Artigues, S. Demassey, and E. Neron, editors, *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*, pages 203–212. John Wiley & Sons, Inc., London, 2008.

B. Denton, J. Viapiano, and A. Vogl. Optimization of surgery sequencing and scheduling decisions under uncertainty. *Health Care Management Science*, 10: 13–24, 2007. ISSN 13869620. doi: 10.1007/s10729-006-9005-4.

L.-E. Drezet and J.-C. Billaut. A project scheduling problem with labour constraints and time-dependent activities requirements. *International Journal of Production Economics*, 112:217–225, 2008. doi: 10.1016/j.ijpe.2006.08.021.

A. Duarte, N. Mladenovic, J. Sánchez-Oro, and R. Todosijevic. Variable Neighborhood Descent. In R. Martí, P.M. Pardalos, and M. G. C. Resende, editors, *Handbook of Heuristics*, chapter 12, pages 341–367. Springer International Publishing, 1st edition, 2018.

S. A. Erdogan and B. Denton. Dynamic appointment scheduling of a stochastic server with uncertain demand. *INFORMS Journal on Computing*, 25(1):116–132, 2013. ISSN 10919856. doi: 10.1287/ijoc.1110.0482.

T. Felberbauer, W. J. Gutjahr, and K. F. Doerner. Stochastic project management: multiple projects with multi-skilled human resources. *Journal of Scheduling*, pages 1–18, 2018. ISSN 1099-1425. doi: 10.1007/s10951-018-0592-y. URL `https://doi.org/10.1007/s10951-018-0592-y`. Available online.

M. Firat and C.A.J. Hurkens. An improved MIP-based approach for a multi-skill workforce scheduling problem. *Journal of Scheduling*, 15:363–380, 2012. doi: 10.1007/s10951-011-0245-x.

C. García-Martínez, F. J. Rodriguez, and M. Lozano. Genetic Algorithms. In R. Martí, P. M. Pardalos, and M. G. C. Resende, editors, *Handbook of Heuristics*, chapter 15, pages 431–464. Springer International Publishing, 1st edition, 2018.

Y. Gocgun. Simulation-based approximate policy iteration for dynamic resource-constrained project scheduling. *Health Care Management Science*, 21(3):317–325, 2018. doi: 10.1007/s10729-016-9388-9.

D. Gupta and B. Denton. Appointment scheduling in health care: Challenges and opportunities. *IIE Transactions*, 40:800–819, 2008.

W. J. Gutjahr and K. A. Froeschl. Project portfolio selection under uncertainty with outsourcing opportunities. *Flexible Services and Manufacturing Journal*, 25:255–281, 2013. ISSN 03772217. doi: 10.1109/TEVC.2011.2132725.

R. Hall. Matching Healthcare Resources and Patient Needs. In R. Hall, editor, *Handbook of Healthcare System Scheduling*, chapter 1, pages 1–9. Springer, New York Dordrecht Heidelberg London, 1st edition, 2012.

E. Hans, G. Wullink, M. Van Houdenhoven, and G. Kazemier. Robust surgery loading. *European Journal of Operational Research*, 185:1038–1050, 2008. doi: 10.1016/j.ejor.2006.08.022.

P. Hansen, N. Mladenovic, and J. A. Moreno Perez. Variable neighbourhood search: methods and applications. *4OR*, 6:319–360, 2008. ISSN 02545330. doi: 10.1007/s10479-009-0657-6.

P. Hansen, N. Mladenovic, J. Brimberg, and J. A. Moreno Perez. Variable Neighborhood Search. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, chapter 3, pages 61–86. Springer, New York Dordrecht Heidelberg London, 2nd edition, 2010.

C. Heimerl and R. Kolisch. Scheduling and staffing multiple projects with a multi-skilled workforce. *OR Spectrum*, 32(2):343–368, 2010. ISSN 01716468. doi: 10.1007/s00291-009-0169-4.

V. Hemmelmayr, V. Schmid, and C. Blum. Variable neighbourhood search for the variable sized bin packing problem. *Computers and Operations Research*, 39(5):1097–1108, 2012. ISSN 03050548. doi: 10.1016/j.cor.2011.07.003. URL `http://dx.doi.org/10.1016/j.cor.2011.07.003`.

W. Herroelen and R. Leus. Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165(2):289–306, 2005. ISSN 03772217. doi: 10.1016/j.ejor.2004.04.002.

P. A. Jensen and J. F. Bard. *Operations Research Models and Methods*. John Wiley & Sohns, Inc., Hoboken, 2003.

A. A. Juan, J. Faulin, S. E. Grasman, M. Rabe, and G. Figueira. A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, 2:62–72, 2015. ISSN 22147160. doi: 10.1016/j.orp.2015.03.001. URL `http://dx.doi.org/10.1016/j.orp.2015.03.001`.

G. C. Kaandorp and G. Koole. Optimal outpatient appointment scheduling. *Health Care Management Science*, 10(3):217–229, 2007. ISSN 13869620. doi: 10.1007/s10729-007-9015-x.

T. Kapamara and D. Petrovic. A heuristics and steepest hill climbing method to scheduling radiotherapy patients. In *Proceedings of the International Conference on Operational Research Applied to Health Services (ORAHS)*, Catholic University of Leuven, Leuven, Belgium, 2009.

T. Kapamara, K. Sheibani, O. Haas, D. Petrovic, and C. Reeves. A review of scheduling problems in radiotherapy. In *Proceedings of the International Control Systems Engineering Conference (ICSE)*, pages 207–211, 2008.

B. Kemper, C. A. J. Klaassen, and M. Mandjes. Optimized appointment scheduling. *European Journal of Operational Research*, 239(1):243–255, 2014. ISSN 03772217. doi: 10.1016/j.ejor.2014.05.027. URL `http://dx.doi.org/10.1016/j.ejor.2014.05.027`.

M. King. *Statistics for process control engineers: A practical approach*. Wiley, 2017. ISBN 9781119383536. doi: 10.1002/9781119383536.

K. J. Klassen and R. Yoogalingam. Improving performance in outpatient appointment services with a simulation optimization approach. *Production and Operations Management*, 18(4):447–458, 2009. ISSN 10591478. doi: 10.1111/j.1937-5956.2009.01021.x.

A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002. ISSN 1052-6234. doi: 10.1137/S1052623499363220. URL `http://epubs.siam.org/doi/10.1137/S1052623499363220`.

P. M. Koeleman and G. M. Koole. Optimal outpatient appointment scheduling with emergency arrivals and general service times. *IIE Transactions on Healthcare Systems Engineering*, 2(1):14–30, 2012. ISSN 19488319. doi: 10.1080/19488300.2012.665154.

R. Kolisch and C. Heimerl. An Efficient Metaheuristic for Integrated Scheduling and Staffing IT Projects Based on a Generalized Minimum Cost Flow Network. *Naval Research Logistics*, 59(2):111–127, 2012. doi: 10.1002/nav.

A. Legrain, M. A. Fortin, N. Lahrichi, and L. M. Rousseau. Online stochastic optimization of radiotherapy patient scheduling. *Health Care Management Science*, 18:110–123, 2015.

P. Leite-Rocha. *Novel approaches to radiotherapy treatment scheduling*. PhD thesis, University of Nottingham, 2011.

E. Lopez. Healthcare supply chains are shifting as cost pressure rise, 2017. URL `https://www.supplychaindive.com/news/hospital-supply-chain-cost-pressure-change/512721/`. Accessed 2019-01-21.

H. R. Lourenço, O. C. Martin, and T. Stützle. Iterated Local Search: Framework and Applications. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, pages 363—-397. Springer, New York Dordrecht Heidelberg London, 2nd edition, 2010.

B. Maenhout and M. Vanhoucke. A hybrid scatter search heuristic for personalized crew rostering in the airline industry. *European Journal of Operational Research*, 206:155–167, 2010. ISSN 0377-2217. doi: 10.1016/j.ejor.2010.01.040. URL `http://dx.doi.org/10.1016/j.ejor.2010.01.040`.

C. Mancilla and R. Storer. A sample average approximation approach to stochastic appointment sequencing and scheduling. *IIE Transactions*, 44(8):655–670, 2012. ISSN 0740817X. doi: 10.1080/0740817X.2011.635174.

R. Martí, J. A. Lozano, A. Mendiburu, and L. Hernando. Multi-start Methods. In R. Martí, P. M. Pardalos, and M. G. C. Resende, editors, *Handbook of Heuristics*, chapter 6, pages 155–176. Springer International Publishing, 2018.

J. Marynissen and E. Demeulemeester. Literature review on multi-appointment scheduling problems in hospitals. *European Journal of Operational Research*, 272:407–419, 2019. ISSN 03772217. doi: 10.1016/j.ejor.2018.03.001. URL `https://doi.org/10.1016/j.ejor.2018.03.001`. In press.

J. Maschler, M. Riedler, M. Stock, and G. R. Raidl. Particle therapy patient dcheduling: First heuristic approaches. In E. K. Burke, editor, *PATAT2016: Proceedings of the 11th International Conference of the Practice and Theory of Automated Timetabling*, pages 223–244, 2016.

J. Maschler, T. Hackl, M. Riedler, and G. R. Raidl. An enhanced iterated greedy metaheuristic for the particle therapy patient scheduling problem. In A. Duarte, A. Viana, J. Angel, B. Mélian, and H. Ramalhinho, editors, *Proceedings of the MIC and MAEB 2017 Conferences*, pages 465–474, Barcelona, 2017a.

J. Maschler, M. Riedler, and G. R. Raidl. Particle therapy patient scheduling: Time estimation for scheduling sets of treatments. In R. Moreno-Díaz, F. Pichler, and

A. Quesada-Arencibia, editors, *Computer Aided Systems Theory – EUROCAST 2017*. Springer International Publishing Switzerland, 2017b.

C. Men. *Optimization models for radiation therapy: Treatment planning and patient scheduling*. PhD thesis, University of Florida, 2009.

E. Naudin, P.Y.C. Chan, M. Hiroux, T. Zemmouri, and G. Weil. Analysis of three mathematical models of the Staff Rostering Problem. *Journal of Scheduling*, 15: 23–38, 2012. doi: 10.1007/s10951-009-0155-3.

F. Neri, C. Cotta, and P. Moscato, editors. *Handbook of Memetic Algorithms*. Springer Berlin Heidelberg, 2012. ISBN 9783642232466. doi: 10.1007/978-3-642-23247-3.

T. Ohno. Particle radiotherapy with carbon ion beams. *The EPMA Journal*, 4(9), 2013. ISSN 1878-5077. doi: 10.1186/1878-5085-4-9.

D. Petrovic, M. Morshed, and S. Petrovic. Genetic algorithm based scheduling of radiotherapy treatments for cancer patients. In *Proceedings of the Conference on Artificial Intelligence in Medicine (AIME)*, volume 5651, pages 101–105, 2009.

D. Petrovic, M. Morshed, and S. Petrovic. Multi-objective genetic algorithms for scheduling of radiotherapy treatments for categorised cancer patients. *Expert Systems with Applications*, 38(6):6994–7002, 2011.

S. Petrovic and E. Castro. A genetic algorithm for radiotherapy pre-treatment scheduling. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6625 LNCS (PART 2):454–463, 2011.

S. Petrovic and P. Leite-Rocha. Constructive approaches to radiotherapy scheduling. *World Congress on Engineering and Computer Science (WCECS)*, pages 722–727, 2008a.

S. Petrovic and P. Leite-Rocha. Constructive and GRASP approaches to radiotherapy treatment scheduling. In *Proceedings – Advances in Electrical and Electronics Engineering – IAENG Special Edition of the World Congress on Engineering and Computer Science 2008, WCECS 2008*, pages 192–200, 2008b.

S. Petrovic, W. Leung, X. Song, and S. Sundar. Algorithms for radiotherapy treatment booking. *25th Workshop of the UK Planning and Scheduling Special Interest Group*, pages 105–112, 2006.

PTCOG. Particle Therapy Co-Operative Group, 2017. URL `https://www.ptcog.ch/index.php/facilities-in-operation`. Accessed 2018-01-17.

C. R. Reeves. Genetic Algorithms. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics2*, pages 109–139. Springer, New York Dordrecht Heidelberg London, 2nd edition, 2010.

M. Samudra, C. Van Riet, E. Demeulemeester, N. Vansteenkiste, and F. Rademakers. Scheduling operating rooms: achievements, challenges and pitfalls. *Journal of Scheduling*, 19:493–525, 2016. ISSN 10946136. doi: 10.1007/s10951-016-0489-6.

A. Sauré, J. Patrick, S. Tyldesley, and M. L. Puterman. Dynamic multi-appointment patient scheduling for radiation therapy. *European Journal of Operational Research*, 223:573–584, 2012.

P. Smet, A. T. Ernst, and G. Van den Berghe. Heuristic decomposition approaches for an integrated task scheduling and personnel rostering problem. *Computers and Operation Research*, 76:60–72, 2016. ISSN 0305-0548. doi: 10.1016/j.cor.2016.05.016. URL `http://dx.doi.org/10.1016/j.cor.2016.05.016`.

B. W. Steward and C. P. Wild. World Cancer Report 2014. Technical report, World Health Organization, International Agency for Research on Cancer, 2014.

T. Stützle and R. Ruiz. Iterated Local Search. In R. Martí, P. M. Pardalos, and M. G. C. Resende, editors, *Handbook of Heuristics*, chapter 19, pages 579–605. Springer International Publishing, 2018.

G. Syswerda. Schedule optimization using genetic algorithms. In L. Davis, editor, *Handbook of Genetic Algorithms*, pages 332–349. International Thomson Publishing Services, London, 1996.

J.-S. Tancrez, B. Roland, J.-P. Cordier, and F. Riane. Assessing the impact of stochasticity for operating theater sizing. *Decision Support Systems*, 55:616–628, 2013. doi: 10.1016/j.dss.2012.10.021.

S. Van De Vonder, E. Demeulemeester, W. Herroelen, and R. Leus. The use of buffers in project management: The trade-off between stability and makespan. *International Journal of Production Economics*, 97(2):227–240, 2005. ISSN 09255273. doi: 10.1016/j.ijpe.2004.08.004.

S. Van De Vonder, E. Demeulemeester, R. Leus, and W. Herroelen. Proactive-reactive project scheduling - trade-offs and procedures. In J. Jozefowska and J. Weglarz, editors, *Perspectives in modern project scheduling*, chapter 2, pages 25—-53. Springer, 2006.

S. Van De Vonder, E. Demeulemeester, and W. Herroelen. A classification of predictive-reactive project scheduling procedures. *Journal of Scheduling*, 10(3): 195–207, 2007. ISSN 10946136. doi: 10.1007/s10951-007-0011-2.

J. Van Den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester, and L. De Boeck. Personnel scheduling: A literature review. *European Journal of Operational Research*, 226(3):367–385, 2013. ISSN 03772217. doi: 10.1016/j.ejor.2012.11.029. URL `http://dx.doi.org/10.1016/j.ejor.2012.11.029`.

C. R. Vela, R. Varela, and M. A. González. Local search and genetic algorithm for the job shop scheduling problem with sequence dependent setup times. *Journal of Heuristics*, 16(2):139–165, 2010. ISSN 13811231. doi: 10.1007/s10732-008-9094-y.

B. Vieira, E. W. Hans, C. Van Vliet-Vroegindeweij, J. Van De Kamer, and W. Van Harten. Operations research for resource planning and -use in radiotherapy: A literature review. *BMC Medical Informatics and Decision Making*, 16(149): 1–11, 2016. ISSN 14726947. doi: 10.1186/s12911-016-0390-4. URL `http://dx.doi.org/10.1186/s12911-016-0390-4`.

P. Vogl and R. Braune. Combined activity selection and skilled staff scheduling for the Red Cross blood donation services. In E. K. Burke, L. Di Gaspero, B. McCollum, N. Musliu, and E. Oezcan, editors, *PATAT 2018: Proceedings of the 12th International Conference of the Practice and Theory of Automated Timetabling*, pages 413–415, 2018. URL `http://www.patatconference.org/patat2018/proceedings/`.

P. Vogl and R. Braune. Heuristic Decomposition Approach for Integrated Activity Selection and Skilled Staff Scheduling. In preparation for Annals of Operations Research, Special Issue: The Practice and Theory of Automated Timetabling, 2019.

P. Vogl, R. Braune, and K. F. Doerner. A multi-encoded genetic algorithm approach to scheduling recurring radiotherapy treatment activities with alternative resources, optional activities, and time window constraints. *Computer Aided Systems Theory – EUROCAST 2017, Lecture Notes in Computer Science*, 10671 (1):373–382, 2018a.

P. Vogl, R. Braune, and K. F. Doerner. Scheduling recurring radiotherapy appointments in an ion beam facility: Considering optional activities and time window constraints. *Journal of Scheduling*, pages 1–18, 2018b. ISSN 10946136. doi: 10.1007/s10951-018-0574-0. URL `https://doi.org/10.1007/s10951-018-0574-0`. Available online.

P. Vogl, R. Braune, W. J. Gutjahr, and K. F. Doerner. Radiotherapy Appointment Scheduling Considering Stochastic Activity Durations. Under revision at Flexible Services and Manufacturing Journal (21.12.2018), 2018c.

C. Walters. Depleting resources adding pressure to healthcare, 2019. URL `https://www.pwc.com/gx/en/industries/healthcare/em`. Accessed 2019-01-22.

C. M. Washington and D. Leaver. *Principles and Practice of Radiation Therapy*. Mosby, 2016.

F. Werner. A Survey of Genetic Algorithms for Shop Scheduling Problems. In P. Siarry, editor, *Heuristics: Theory and Applications*, chapter 8, pages 161–222. Nova Science Publishers, New York, 2013.

World Health Organization. GLOBOCAN 2012, 2012. URL `http://globocan.iarc.fr/Default.aspx`. Accessed 2017-08-03.

G. Zäpfel, R. Braune, and M. Bögl. *Metaheuristic Search Concepts, A Tutorial with Applications in Production and Logistics*. Springer, Heidelberg, 2010.

Abbreviations

A1G	... Algorithm Variant 1 with Greedy Re-start (Chapter 8)
A1R	... Algorithm Variant 1 with Random Re-start (Chapter 8)
A1S	... Algorithm Variant 1 with Shaking (Chapter 8)
A2G	... Algorithm Variant 2 with Greedy Re-start (Chapter 8)
A2R	... Algorithm Variant 2 with Random Re-start (Chapter 8)
A2S	... Algorithm Variant 2 with Shaking (Chapter 8)
cGAILS	... Combined Genetic Algorithm and Iterated Local Search (Chapter 4)
DET	... Deterministic Solution Evaluation Technique (Chapter 5)
DT	... Daily Treatment (Part I)
FDT	... First Daily Treatment (Part I)
FS	... Finish-Start (Part I)
GA	... Genetic Algorithm
GRASP	... Greedy Randomized Adaptive Search Procedure
ILS	... Iterated Local Search
LB	... Lower Bound
LDT	... Last Daily Treatment (Part I)
MIP	... Mixed Integer Programming
OS	... Offspring Selection
PBX	... Position-Based Crossover
PET	... Positron Emission Tomography (Part I)
RO	... Radio-Oncologist (Part I)
RPSP	... Radiotherapy Patient Scheduling Problem (Part I)
SCOP	... Stochastic Combinatorial Optimization Problem
STO	... Stochastic Solution Evaluation Technique (Chapter 5)
VND	... Variable Neighborhood Descent
WCE	... Weekly Control Examination (Part I)
WTE	... Waiting Time Estimation Solution Evaluation Technique (Chapter 5)

List of Figures

List of Tables