Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Operating-Room Scheduling Modeled as a Non-Cooperative Game

Ondřej Tkadlec

Supervisor: Doc. Ing. Přemysl Šůcha, Ph.D.
May 2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Tkadlec**	Jméno:	**Ondřej**	Osobní číslo:	**474419**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Datové vědy**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Rozvrhování chirurgických operaci modelované jako nekooperativní hra

Název diplomové práce anglicky:

Operating-room scheduling modeled as a non-cooperative game

Pokyny pro vypracování:

Effective decision-making in the operating room department is hampered due to conflicting interests of stakeholders. The aim of this thesis is to study a multi-agent scheduling problem to distribute the available operating room time between surgeons, which is interrelated with the planning of surgical cases. The assignment defines the following tasks:
1) analyze the data on the performed surgical operations,
2) review the existing literature,
3) design and implement the scheduling algorithm,
4) benchmark the algorithm on synthetic and on real data and compare it with existing approaches using the price of anarchy and price of stability

Seznam doporučené literatury:

[1] Šůcha, P., Agnetis, A., Šidlovský, M., and Briand, C. (2021). Nash equilibrium solutions in multi-agent project scheduling with milestones. European Journal of Operational Research, 294(1):29–41.
[2] Milicka, P., Šůcha, P., Vanhoucke, M., and Maenhout, B. (2022). The bilevel optimisation of a multi-agent project scheduling and staffing problem. European Journal on Operational Research, 1(296):72–86.
[3] Agnetis, A., Coppi, A., Corsini, M., Dellino, G., Meloni, C., and Pranzo, M. (2014). A decomposition approach for the combined master surgical schedule and surgical case assignment problems. Health Care Management Science, 17(1):49–59.
[4] Savva, N. and Keskinocak, P. (2019). A review of the healthcare-management (modeling) literature published at Manufacturing and Service Operations Management. Manufacturing and Service Operations Management, 22(1):59–72.

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Přemysl Šůcha, Ph.D.   katedra řídicí techniky   FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **20.02.2023**   Termín odevzdání diplomové práce: _____________

Platnost zadání diplomové práce: **16.02.2025**

doc. Ing. Přemysl Šůcha, Ph.D.	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce		podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.	
Datum převzetí zadání	Podpis studenta

Acknowledgements

I would like to express my sincere gratitude to my supervisor Doc. Ing. Přemysl Šůcha, Ph.D. for his invaluable guidance, encouragement and time spent during our cooperation. His expertise and constant feedback have been instrumental in shaping my thesis.

I would also like to extend my heartfelt thanks to my family for their unwavering love and support. Their constant encouragement and belief in me have been a source of strength and motivation.

Last but not least, I want to thank my girlfriend for her patience, understanding and encouragement during this challenging journey.

I am grateful for the support and encouragement of these individuals, without whom this thesis would not have been possible.

Declaration

I declare that I worked on this thesis independently and that all sources of information used in this thesis have been duly acknowledged.

Furthermore, I confirm that I have adhered to the ethical principles governing academic research.

In Prague, May 20, 2023

Abstract

This diploma thesis deals with problem of multi-agent operating room scheduling. The goal is to assign available operating room blocks to surgeons and simultaneously assign patients to these blocks. Even though the head of surgeon group and individual surgeons can be both looked upon as stakeholders with conflicting objectives, they are together responsible for surgery planning. The hierarchical nature of the problem motivates the use of bilevel optimization.

The problem can be grasped as a non-cooperative zero-sum game, because no surgeon can improve his objective without worsening the objective of other surgeons (removing the OR time). Two ways to tackle this problem are proposed. Firstly, the problem is formulated as an integer linear program (ILP). In the second approach, a dedicated branch-and-price algorithm is proposed. In addition, we perform a data analysis of a real-life dataset provided by University Hospital of Hradec Králové and an artificial (publicly available) dataset.

In the computational experiments, the integer linear program and the branch-and-price are validated under different scenarios, parameters and with various speed-up mechanisms. Finally, the algorithm is tested on the artificial and the real-life dataset.

Keywords: Operating room scheduling, Bilevel optimazation, Integer linear programming, Branch-and-price

Supervisor: Doc. Ing. Přemysl Šůcha, Ph.D.
Technická 2
160 00 Praha 6

Abstrakt

Tato diplomová práce se zabývá problémem rozvrhování chirurgických operací. Cílem je přiřadit dostupné bloky operačních sálů chirurgům a současně přiřadit pacienty do těchto bloků. I když vedoucí chirurgů a jednotliví chirurgové mohou být považováni za zainteresované strany s protichůdnými cíli, společně jsou zodpovědní za plánování operací. Tahle hierarchická povaha problému motivuje použití dvouúrovňové optimalizace.

Problém lze chápat jako nekooperativní hru s nulovým součtem, protože žádný chirurg nemůže zlepšit svoji výplatní funkci, aniž by zhoršil výplatní funkci ostatních chirurgů. Pro tento problém byly navrženy dva algoritmy. V prvním je problém formulován jako celočíselný lineární program. Druhý algoritmus je dedikovaný branch-and-price. Dále je provedena datová analýza z dat poskytnutých Fakultní nemocnicí Hradec Králové a umělého (veřejně dostupného) datasetu.

Ve výpočetních experimentech je porovnán celočíselný lineární program a branch-and-price v různých scénářích, s různými parametry a různými mechanismy zrychlení. Nakonec je algoritmus otestován na reálných i umělých datech.

Klíčová slova: Rozvrhování operací, Dvouúrovňová optimalizace, Celočíselné lineární progrmování, Branch-and-price

Překlad názvu: Rozvrhování chirurgických operaci modelované jako nekooperativní hra

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Motivation

Efficiency of operating room (OR) utilization has been of high interest in recent decades. Operating rooms are the most costly out of all facilities in a hospital, comprising of up to 33 % of a hospital's expenditure (Macario et al., 1995). However, OR scheduling is a complex task that involves multiple factors, such as surgical priorities, surgeon availability and patient characteristics, making it a challenging problem that requires careful consideration and planning. The effective utilization of operating rooms leads to both increasing the revenues by the number of treatments/surgeries carried out and decreasing the costs by reducing idle time between patients. In real life environment, schedules are created in ad hoc, often very ineffective manner, which may lead to underutilization and discontent with the schedule from all parties involved.

Additionally, the decision-making may be impeded due to conflicting interests of stakeholders. To tackle this problem, a bilevel approach is typically used. Each level/stage has different objective and the corresponding solution is passed to the next level as a constraint, creating a final solution. This solution can be understood as an agreement of both sides. In our case, the central authority is a head of the surgical department, also referred to as the *head of surgeons* or the *leader*, whose objective is to maximize the OR utilization and the number of patients treated, possibly weighted by individual priorities of each patient. These priorities take into consideration such aspects as waiting time, duration or revenue of individual cases. Other stakeholders are the individual surgeons, also referred to as the *followers*. Surgeon's objective is to assign such patients so that it complies with his objective, which is typically different from the leader's one. The surgeons are assumed to behave in a selfish way, meaning that they do not care about the utilization from the leader's perspective.

All these factors motivate us to create a bilevel framework for OR scheduling, which takes interests of all agents into consideration.

This thesis is a collateral work to the paper (Maenhout et al., 2023), of whom the author of this thesis is part. The notation, models and experiments are patterned after the paper.

1.2 Contribution

In this thesis, we concentrate on multi-agent OR scheduling through block assignment framework. The leader creates and distributes OR blocks, which are defined by starting time and length. Within these OR blocks, individual surgeons strive to stack patients. In the thesis, we focus on short-term scheduling, which is typically a time horizon of one or two weeks. The leader has access to a list of patients and their characteristics: what surgeon they belong to and their duration. Another characteristics of a patient are his priorities. In general, we assume two types of priorities per patient - leader's priority (sometimes called penalty for not performing operation of the patient) and follower's priority. If these are not provided, they can be consider equal for all patients or for testing purposes, we can determine them randomly.

Below, the main contributions of the thesis are listed:

- We formally define the problem of OR scheduling with respect to our case.
- We introduce an integer linear programming (ILP) model that serves to address our problem as a baseline solution.
- We design and implement a dedicated branch-and-price algorithm to solve our problem and outperform ILP model in specific cases.
- We use lazy constraints to ensure bilevel optimality of our solution.
- We present several speed-up techniques in order to improve the performance and computational time.
- Experiments are run on a real dataset (artificial data are used as well).

1.3 Outline

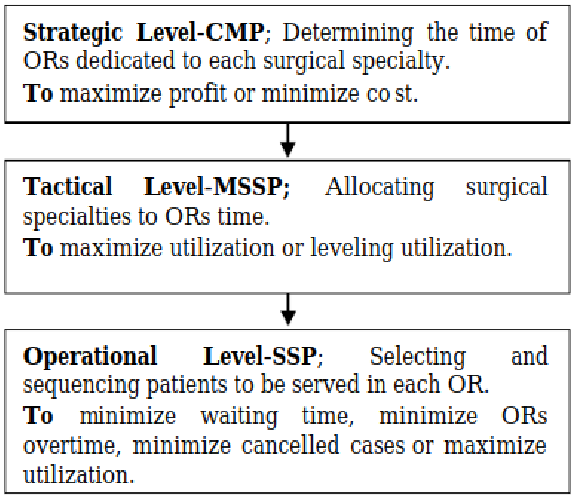
The thesis is organized in the following way. In section 2, we discuss relevant literature related to OR scheduling and bilevel optimization. The general branch-and-price algorithm and his fundamentals (master problem, pricing problem, column generation) are described in section 3. The problem is formulated in section 4, where we present various attributes, parameters and assumptions. Integer linear programming model is formulated in section 5. We introduce variables and constraints used in ILP. First, a mathematical model of the leader's problem is presented followed by the follower's problem. The concept of callback functions is introduced together with two types of lazy constraints. The main branch-and-price algorithm is presented in section 6. We first formulate the Dantzig-Wolfe decomposition including master problem, dual master problem and subproblem along with variables and constraints used within the model. Then, we alter the lazy constraints from ILP to suit the branch-and-price definition. In this section, we also generate initial patterns. We first show the empty patterns, then an initial heuristic is presented to generate more sophisticated initial columns. At the end of the section, we discuss a speed-up technique called lazy constraint remembering and finally, we emphasize the importance of dummy pattern generation. In section 7, we analyse the data from University Hospital of Hradec Králové. We verify that they satisfy some characteristics and assumptions stated in the literature. Experimental results are presented in section 8.

Chapter 2
Related work

2.1 Surgery Planning

The topic of surgery planning (OR scheduling) has been widely studied over the past years. The motivation is simple - OR facilities are the biggest source of revenue and cost (Association et al., 2003). The problem became even more popular during the COVID-19 pandemic (Momeni et al., 2022). Similar problem of integrated recovery room planning and scheduling during the pandemic was studied in (Chaieb et al., 2022).

Operating room scheduling is often approached on three levels: strategic (long-term), tactical (medium-term) and operational (short-term) (Rahimi and Gandomi, 2020). These three levels are more described in (Maenhout et al., 2023).



Strategic Level-CMP; Determining the time of ORs dedicated to each surgical specialty.
To maximize profit or minimize co st.

Tactical Level-MSSP; Allocating surgical specialties to ORs time.
To maximize utilization or leveling utilization.

Operational Level-SSP; Selecting and sequencing patients to be served in each OR.
To minimize waiting time, minimize ORs overtime, minimize cancelled cases or maximize utilization.

Figure 2.1: Three levels of OR scheduling. Figure taken from (Abdelrasol et al., 2013)

Authors in (Wullink et al., 2007) examined the best way to reserve OR time for emergency surgery. They used two approaches of reserving capacity. First, concentrating all reserved OR capacity in dedicated emergency operating rooms, and second, evenly reserving capacity in all elective operating rooms. In this thesis, the emergency surgeries are omitted.

Many papers study the problem of surgeon-block assignment and patient scheduling separately. Some papers take into consideration the preferences of surgeons or other staff (Meskens et al., 2013)(Ahmed and Ali, 2020). OR scheduling and planning where blocks are assigned to surgeons and patients/surgical cases are planned within each OR block during weekly time horizon is discussed in (Mazloumian et al., 2022)(Huele and Vanhoucke, 2014).

Whereas this thesis assumes deterministic parameters such as duration of the patient's surgery, other papers approach the problem in stochastic way through simulations (Persson and Persson, 2009)(Jebali and Diabat, 2015). As for the exact and heuristic methods, applications of mixed integer linear programming are very common (Maaroufi et al., 2016)(Ma et al., 2022). Column generation method has also been popular amongst authors in order to efficiently find surgery schedules (Doulabi et al., 2016)(Kamran et al., 2020). In (Ahmed and Ali, 2020), the authors propose a branch-and-price algorithm with different speed-up techniques. Branch-and-price is also used in (Cardoen et al., 2009), where authors compare their algorithm with mixed integer linear programming solution and examine several branching strategies. They also use speed-up techniques e.g. Lagrangian bound, initial heuristics and column elimination.

As there are hundreds of papers regarding OR scheduling and planning in different forms, we refer to some overview papers (Harris and Claudio, 2022)(Cardoen et al., 2010)(Zhu et al., 2018).

2.2 Bilevel Optimization

Bilevel optimization is a special type of optimization that models real-world problems with two decision-makers (in game theory often called agents), who have a hierarchical relationship and possibly conflicting interests. In this sense, this framework can be look upon as a Stackelberg game (Von Stackelberg, 2010). The Stackelberg game includes two types of players - a leader and a follower/followers. In our case, the head of surgeons (leader) is in the upper level of hierarchy whereas the individual surgeons (followers) are in the lower level. The leader makes a decision that is passed onto the followers. Then, the followers react and send their responses back to the leader, who can eventually adjust his decision. As for the complexity, bilevel optimization is known to be strongly $\mathcal{NP}$-hard (Hansen et al., 1992).

The applications of bilevel optimization are vast. In (Whittaker et al., 2017), authors deal with an environmental issue. The leader is an agency that has environmental objectives and the followers are producers who try to maximize their profit. In another example, authors in (Sinha et al., 2013) discuss the problem of designing a tax policy. The leader in this case is the regulating authority, and it tries to maximize its total tax revenue over multiple periods while trying to minimize the environmental damages caused by a mining company. The follower is the mining company whose sole objective is to maximize its total profit over multiple periods under the limitations set by the leader. Other applications of bilevel optimization regard e.g. toll setting problem (González Velarde et al., 2015), chemical industry (Clark and Westerberg, 1990) and defense industry, where often a attacker/defender (leader/follower) Stackelberg game is considered (Scaparra and Church, 2008)(Brown et al., 2006).

There are several methodologies to solve bilevel optimalization problems. In (Bard and Moore, 1992), authors claim that some fathoming rules of branch-and-bound algorithm used in mixed integer linear programming cannot be applied in the bilevel context, like fathoming when the relaxed subproblem is worse than the value of the incumbent solution or fathoming when the solution of the relaxed subproblem is feasible for the mixed integer linear problem. Another method for solving bilevel problems with integer variables in the upper level is via Benders decomposition (Saharidis and Ierapetritou, 2009). Recent paper (Milička et al., 2022) suggests solving bilevel problems by combining mathematical programming and lazy constraint generation. When an integer solution is found, the optimality of follower's problem is checked. Lazy constraints are added via callback functions.

Chapter 3

Branch and Price

3.1 Overview and Motivation

Branch and price (branch-and-price) is an algorithm of combinatorial optimization for solving integer linear programming and mixed integer linear programming problems. It's an extended version of the branch-and-bound algorithm that is common for solving ILP problems. Branch-and-price is suitable for problems that contain huge number of variables, typically a number that is exponential relative to the input needed to describe the original problem.

As stated in (Easton et al., 2004), there are several reasons why an ILP formulation with a high number of variables may be convenient in comparison to the conventional compact formulation when the solution methodology is LP-based branch-and-bound. First, branch-and-bound effectiveness is highly dependent on the strength of the LP relaxation. Weak LP relaxations lead to excessive branching and long computational times. By allowing a huge number of variables, we can often create formulations that give stronger LP bounds. Second, huge formulations can encapsulate difficult modeling issues in the definition of the variables.

For instance, our basic ILP formulation of OR scheduling problem involves two variables and high number of constraints. If a variable for every feasible schedule is created, the final formulation of the master problem is much simpler with only one variable and lower number of constraints.

On the other hand, the number of possible OR schedules is huge. Branch-and-price doesn't use all of these variables, but starts with only a small subset of the feasible columns in order to reduce the computational and memory requirements. Columns are generated and added to the model throughout the search tree to find an optimal solution to the full integer program. That is why branch-and-price is a hybrid of branch-and-bound and column generation.

3.2 Algorithm Description

The original problem is typically reformulated (e.g. by using Dantzig-Wolfe decomposition) into a master problem and a subproblem. Switching between the master problem and the subproblem until no new column with negative reduced cost is found is called column generation. After the column generation is finished, the integrality of the solution is checked. If the solution is integral and its objective value is better than the incumbent solution, then a new incumbent solution is set. If the integrality condition is fullfilled, the node is fathomed. Otherwise, the algorithm branches. When there are no more nodes to visit, the algorithm terminates and the optimal solution is the incumbent solution.

The branch-and-price does not specify how to deal with branching strategy. In our case, we branch over the surgeon-block assignment. In general, the branching strategy may be arbitrary, but must be chosen with regard to efficiency and computational time.

The master problem uses variables corresponding to individual columns. The restricted master problem (RMP) contains only a small subset of columns. The initial subset can be typically generated via some heuristics, or empty columns can be added (in case of OR scheduling, for every surgeon we assign no blocks). In branch-and-price algorithm, a LP relaxation of RMP is always solved. Value of a variable indicates "how much" the column has been selected. When the master problem is solved, the dual variables are used in the subproblem to determine the optimality of the solution. There are usually very few constraints in the master problem as most the constraints are built in the columns generated in the subproblem.

The subproblem (also pricing problem) serves to add columns to the master problem or decide that no new columns can be added. If we can generate a column with negative reduced cost (in case of minimization problem), such column is appended to the basis and the master problem is reoptimized. Multiple columns can be generated and added per iteration. In our problem, the subproblem is solved for each surgeon $s \in S$ independently, meaning that up to $|S|$ new columns can be added per iteration. In experiments 8, we compare different speed-up mechanisms. One of them is adding single or multiple patterns per iteration.

In each node of the search tree, we perform an algorithm called *column generation*. As stated earlier, column generation takes an initial set of columns and tries to generate new columns that would enter the basis for RMP. In our case, if we fail to generate at least one new column for any of the surgeons, the algorithm stops. The column generation is explained in the pseudocode below.

Algorithm 1 Column Generation function with initial columns K

```
function ColumnGeneration($K$)
    repeat
        $dualValues \leftarrow$ masterProblem($K$)
        $columnsGenerated, newColumns \leftarrow$ subproblem($dualValues, K$)
        $K \leftarrow K \cup newColumns$
    until $columnsGenerated$ is True
    return $K$
```

In the next figure is a flowchart that represents a general branch-and-price algorithm.

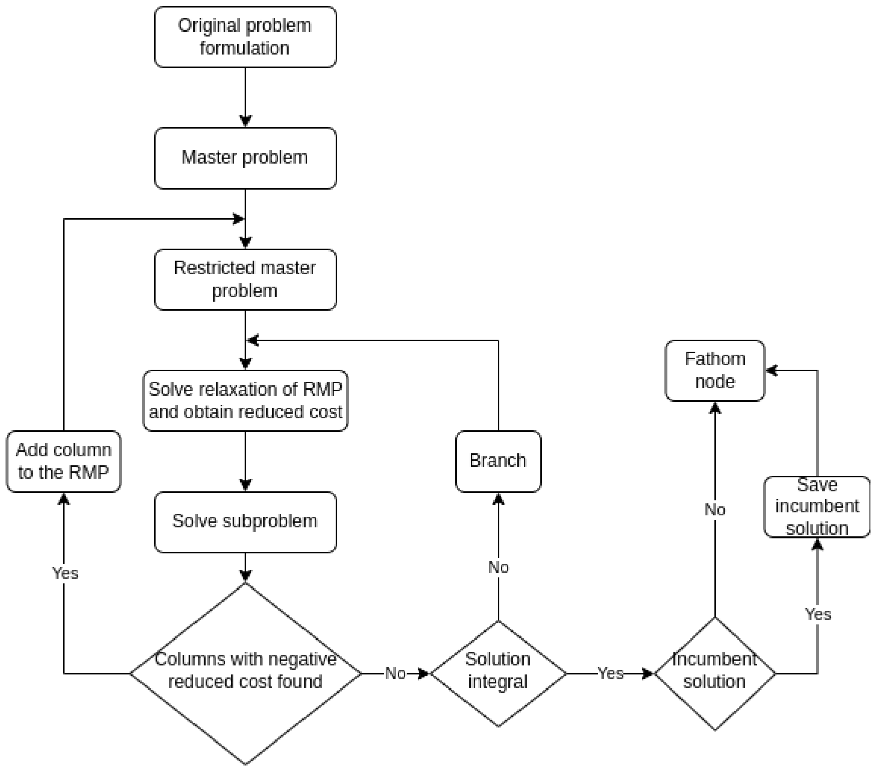


Figure 3.1: Diagram of branch-and-price algorithm. Adapted from (Ponboon et al., 2016)

Chapter 4

Problem Statement

The problem is characterized as OR scheduling on an operational level, meaning we seek to obtain an bilevel optimal solution by assigning blocks to surgeons and patients to blocks. An OR block is defined by its start time, end time and the day it is executed. The OR blocks can also be designed with regard to the number of the operating room. It means that an OR block is tied to a specific operating room.

The decision-making within the OR scheduling has a hierarchical structure, which can be thought of as a non-cooperative Stackelberg game. The participants in this game are the head of surgeons, who is hierarchically higher-ranked, and the individual surgeons, who are lower-ranked. The surgeon head is in charge of the scheduling and allocation of OR capacity to the individual surgeons, whereas the individual surgeons are concerned with creating their patient schedule given the allocated capacity.

4.1 Characteristics and Parameters

The head of surgeons considers a short-term surgery schedule, that is typically planned for 1 week (5 working days) or 2 weeks (10 working days). The time horizon contains a set of days D. We consider a specialized surgeon group (certain discipline) that is composed of a set of surgeons S. Each surgeon $s \in S$ has a set of patients P_s in his waiting list that he wants to schedule in the upcoming time horizon. The set of all the patients is denoted by P. It is essential to set a time granularity for this problem. We set this parameter to be a quarter of an hour, meaning all durations of a surgery must be a multiple of 15 minutes. The surgeons can operate in a set of operation rooms R. Given the scheduling time horizon, we can create a set of blocks B (index b), which are defined by start time, end time and the day they are executed. The start and end time determine the block's length (also referred as duration) D_b that has to be a positive integer. For our problem, we consider all operating rooms to be accessible for 8 hours a day. Given the time granularity of 15 minutes, the length of a day is 32 time units. It is also the length of the longest block. A block's time granularity is 2 hours, so a block can start at times 0, 8, 16, 24 and it can terminate at times 8, 16, 24, 32 (note that block's length must be strictly positive). At any time t and day d, we can find overlapping blocks.

This set is denoted as O_{dt}. For further applications, we establish a set of all blocks on day d as B_d.

Next figure is an illustration of all blocks on day $d = 0$. In other words, it represents a set $B_{d=0}$. Only one room is considered in this example and there are 10 blocks in total. To facilitate the demonstration of overlapping blocks at $d = 0$ and $t = 16$, the vertical red dashed line has been added into the figure. All blocks that either start or are in process on this day and time belong to the $O_{d=0,t=16}$ set. Whether block b is in process on day d and time t is also defined by a binary parameter s_{bdt}.

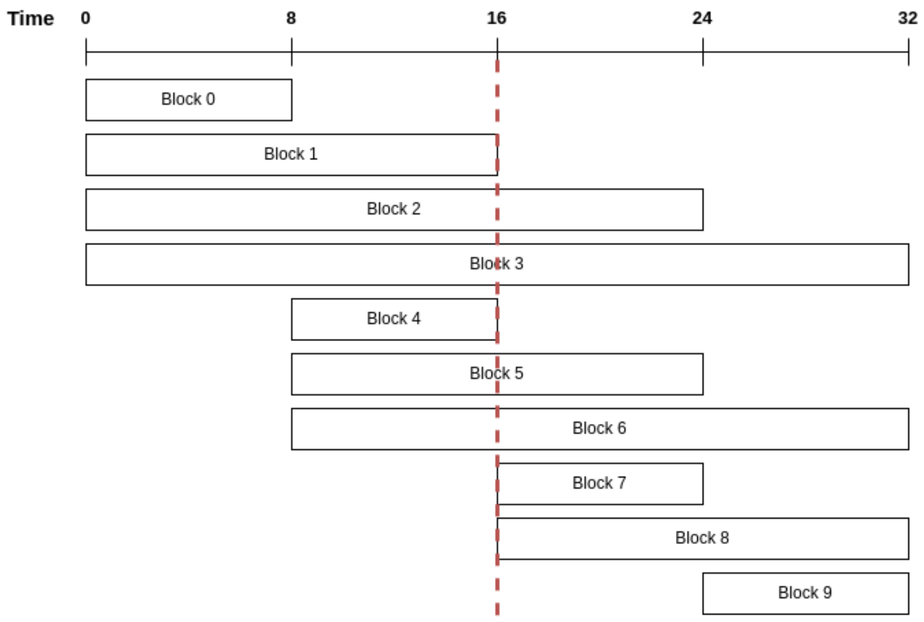


Figure 4.1: All possible blocks in day $d = 0$

Individual patients also have their parameters. Each patient's p surgery is expected to last δ_p. Both the head of surgeons and the surgeon of the patient p perceive differently how important is to perform the patient's surgery. From the leader's point of view, the priority can be looked upon as penalty for not performing the surgery. We denote this priority as $pr_p^{LP} \in \mathbb{R}$. The follower receives a reward $\pi_p \in \mathbb{R}$ for performing a surgery of the patient p.

Leader's objective is composed of two terms. The first one tries to minimize the idle time within the blocks allocated to the surgeon group, which corresponds to maximizing the OR utilization. The second term is the priority penalty associated with patients that are not assigned in the surgical case planning. We can opt for prioritizing one of these terms over the other, which is done by scaling the term by constants α and β. By default, these constants are equal. We also introduce overall capacity of operating rooms $C = |R| \times |W| \times 32$ (length of a day is 32). The leader can choose not to assign any blocks to certain surgeons. This may be perceived as unfair from the surgeon's perspective. For this reason, there can be a common agreement that a surgeon must be assigned at least m blocks. By default, this parameter is set to zero. It is presupposed that the overall capacity of all patients available for the schedule is higher than the capacity of operating rooms. The ratio of the sum of duration of all patients and the overall capacity is referred to as *load*.

$$load = \frac{\sum_{p \in P} \delta_p}{C}$$

All parameters with their indexes are summarized in a table below.

Parameter	**Meaning**
$s \in S$	surgeons
$p \in P$	patients
$p \in P_s$	patients of surgeon s $(P = \bigcup_{s \in S} P_s)$
$d \in W$	days in a week
$t \in T$	possible start times of a block in a day
$r \in R$	available operating rooms
$b \in B$	blocks
$b \in B_d$	available OR blocks in day d $(B = \bigcup_{d \in W} B_d)$
$l \in L$	set of all block durations
$O_{dt} \subseteq B$	set of blocks overlapping at time t on day d
$K_l = \|\{b \in B : D(b) = l\}\|$	number of blocks having length l
$\delta_p > 0$	surgery duration of patient p
$\pi_p \geq 0$	patient's priority from surgeon's perspective
$pr_p^{LP} \geq 0$	patient's priority from leader's perspective
$D_b > 0$	duration of block b
C	overall capacity of operating rooms
α, β	coefficients of the leader's objective function
m	minimum number of assigned block per surgeon

4.2 Assumptions

In this section, the general constraints and assumptions for the model are formulated. These assumptions are later transformed into constraints in the ILP and branch-and-price model. The general assumptions are in the following list.

1. A block in certain room can only be assigned to at most one surgeon.
2. Assignment of blocks can only be in a consecutive manner.
3. A surgeon can not be assigned into two different operating room in one day.
4. A patient can be operated only once.
5. If a surgery started it can not be interrupted with other surgeries, or transferred to another operating rooms.

Chapter 5

Integer Linear Programming Model

Solving the problem of OR scheduling through integer linear programming has been widely studied for example in (Aringhieri et al., 2015) or (Li et al., 2015). In this section, we will introduce variables and constraints related to the model, followed by mathematical formulation of the model. For the sake of bilevel optimization, we introduce a two stage approach that is based on adding lazy constraints to the ILP model (Maenhout et al., 2023).

5.1 Mathematical Formulation of ILP Model

5.1.1 Variables

For the ILP model, we present three binary variables. In the leader's problem, the head of surgeons assigns OR blocks to individual surgeons and thus determines the allocated capacity and the schedule for the individual surgeons. Binary variable y_{sb} equals 1 if block b is assigned to surgeon s and 0 otherwise. Both the head of surgeons and the individual surgeons also try to assign patients into these blocks (although with different objective). Thus, we introduce variable x_{pb} indicating that patient p is assigned to block b. In order to make the final solution bilevel optimal, we define a variable $q_{slk} = 1$ if surgeon s has k blocks of length l and 0 otherwise.

- $x_{pb} \in \{0,1\}$ patient p is allocated to block b (followers' decision)
- $y_{sb} \in \{0,1\}$ block b is assigned to surgeon s (leader's decision)
- $q_{slk} \in \{0,1\}$ the number of blocks of length exactly l assigned to surgeon s is exactly k e.g. if a surgeon s is assigned 3 blocks of length 4, then $q_{s40} = 0,\ q_{s41} = 0,\ q_{s42} = 0,\ q_{s43} = 1,\ q_{s44} = 0,\ \dots$

5.1.2 Constraints

In this section, the general constraints for the model are formulated. Some of the constraints can be derived directly from the assumptions in section 4.2. The constraints, which are later expressed mathematically in describing the ILP and branch-and-price model, are listed below.

1. Capacity of operating rooms cannot be exceeded. At every time t and day d, the maximum number of overlapping blocks that are scheduled cannot be greater than the number of operating rooms.
2. A surgeon can be assigned at most one block a day.
3. A patient can be scheduled only once.
4. Capacity of a block b assigned to surgeon s cannot be exceeded. Thus, the overall duration of patients placed in the block b must not surpass the block's length.
5. Every surgeon must have at least m blocks assigned.

5.1.3 Leader's Problem

$$\begin{aligned}
\min \quad & \alpha(C - \sum_{p,b} \delta_p x_{pb}) + \beta \sum_{p \in P} pr_p^{LP}(1 - \sum_{b \in B} x_{pb}) && (5.1)\\
s.t. \quad & && \\
\forall d \in W, \forall t \in T: \quad & \sum_{b \in O_{dt}} \sum_{s \in S} y_{sb} \leq |R| \qquad \text{capacity of ORs} && (5.2)\\
\forall s \in S,\ \forall d \in W: \quad & \sum_{b \in B_d} y_{sb} \leq 1 \qquad \text{a surgeon can be assigned at most one block a day} && (5.3)\\
\forall s \in S: \quad & \sum_{b \in B} y_{sb} \geq m \qquad \text{the minimum number of the assigned blocks} && (5.4)\\
\forall s \in S, \forall l \in L: \quad & \sum_{k=0}^{|W|} k q_{slk} = \sum_{b \in B:\ D_b = l} y_{sb} \qquad \text{first auxiliary constraint} && (5.5)\\
\forall s \in S, \forall l \in L: \quad & \sum_{k=0}^{|W|} q_{slk} = 1 \qquad \text{second auxiliary constraint} && (5.6)\\
\forall p \in P: \quad & \sum_{b \in B} x_{pb} \leq 1 \qquad \text{a patient can be scheduled only once} && (5.7)\\
\forall s \in S, b \in B: \quad & \sum_{p \in P_s} \delta_p x_{pb} \leq D_b y_{sb} \qquad \text{capacity of the block } b \text{ assigned to surgeon } s \text{ cannot be exceeded} && (5.8)
\end{aligned}$$

5.1.4 Follower's Problem

In the follower's problem, it's necessary to create a set of blocks assigned to surgeon s in the leader's problem. We will refer to this set as BA_s. We will also use variable x'_{pb} in order not to confuse it with x_{pb} from leader's problem.

$$BA_s = \{b \in B : y_{sb} = 1\} \qquad (5.9)$$

Then we define the follower's problem as follows:

$$\forall s \in S: \quad \max\ f_s^{optim} \qquad \text{follower's objective for surgeon } s \in S \tag{5.10}$$

$$s.t.$$

$$\forall s \in S: \quad f_s^{optim} = \sum_{p \in P_s} \pi_p \sum_{b \in BA_s} x'_{pb} \qquad \text{profit of surgeon } s \in S \tag{5.11}$$

$$\forall p \in P: \quad \sum_{b \in BA_s} x'_{pb} \leq 1 \qquad \text{a patient can be scheduled only once} \tag{5.12}$$

$$\forall s \in S, b \in BA_s: \quad \sum_{p \in P_s} \delta_p x'_{pb} \leq D_b \qquad \text{capacity of the block } b \text{ assigned to surgeon } s \text{ cannot be exceeded} \tag{5.13}$$

5.2 Lazy Constraints

Whereas the leader's problem is addressed in the main ILP model, the follower's problem is solved in a callback function. Callbacks are called whenever an event occur, e.g. an integer solution is found. They are available in most of modern MILP solvers.

In section 5.1.3, two more constraints appear in comparison with the constraints in section 5.1.2. Constraint 5.5 assure that the variable q_{slk} contains exact information about the number of blocks of length l surgeon s has been assigned. The second constraint ensures that the number of blocks of length l of surgeon s is unique. The upper index of the sum in both of the constraints comes from the fact that each surgeon can be assigned at most one block per day, so in the extreme case, he can be assigned at most $|W|$ blocks of one length.

In the callback, we first retrieve the integer solution from the leader's problem. We will further use the following notation:

- c - counter of solutions found by the ILP (the leader's problem only)
- $y_{sb}^{(c)}$ is the c-th surgeon-block allocation found by the leader's problem
- $x_{pb}^{(c)}$ is the c-th patient-block assignment found by the leader's problem
- $l \in L$ a set of possible block lengths
- $n_{sl}^{(c)} = \sum_{b \in BA_s : D_b = l} y_{sb}^{(c)}$ is the number of blocks of duration l assigned to surgeon s related to solution c

We will present two types of lazy constraints, which are used to assure the bilevel optimality.

5.2.1 Objective Value Based Lazy Constraint

First lazy constraint is based on comparing the value of the follower's objective function from both the leader's (f_s) and the follower's (f_s^{optim}) perspective. After an instance of assigned blocks is generated by solving the leader's problem, each follower's objective value f_s^{optim} is found and compared to the f_s value in the current solution of the leader's problem.

$$f_s = \sum_{p \in P_s} \pi_p \sum_{b \in BA_s} x_{pb}^{(c)} \tag{5.14}$$

If $f_s < f_s^{optim}$ following lazy constraint is added:

$$\sum_{p \in P_s} \pi_p \sum_{b \in B} x_{pb} + M \cdot \left(|L| - \sum_{l \in L} q_{sln_{sl}^{(c)}} \right) \geq f_s^{optim} \tag{5.15}$$

5.2.2 Assigned Patients Based Lazy Constraint

Second lazy constraint is based on forcing/imposing specific patients if surgeon s is assigned particular set of blocks. As in the previous lazy constraint, we compare the value of f_s and f_s^{optim}. If $f_s < f_s^{optim}$, we retrieve patients of surgeon s assigned in the callback (follower's problem) and call them PA_s.

$$PA_s = \{p \in P_s : \sum_{b \in BA_s} x'_{pb} = 1\} \tag{5.16}$$

The idea is to force the patients $p \in PA_s$ into the schedule. It means that whenever surgeon s is assigned the same composition of blocks (given by variable q_{slk}), he must assign the patients $p \in PA_s$ and must not assign the patients $p \in P_s \setminus PA_s$. The final lazy constraint is as follows:

$$\sum_{p \in PA_s} \sum_{b \in B} x_{pb} - \sum_{p \in P_s \setminus PA_s} \sum_{b \in B} x_{pb} + M \cdot \left(|L| - \sum_{l \in L} q_{sln_{sl}^{(c)}}) \right) \geq |PA_s| \tag{5.17}$$

Chapter 6
Branch And Price Model

The main approach in this thesis for OR scheduling is the branch-and-price. Even though the ILP formulation is more straightforward and is easier to understand and implement, is has several drawbacks, especially on the performance level.

6.1 Dantzig-Wolfe Decomposition

As stated in the section 3.2, the original model is reformulated using Dantzig-Wolfe decomposition into the master problem and the subproblem (Maenhout et al., 2023). We no longer use variables y_{sb} and x_{pb} in the problem statement, but we introduce a new variable θ_s^k. In the master problem, θ_s^k is continuous variable. The variable has the following meaning:

$$\theta_k^s = 1 \Leftrightarrow \text{schedule } k \text{ of surgeon } s \text{ is selected} \tag{6.1}$$

The schedule k of a surgeon s is a vector $\mathbf{o}_s^k \in \{0,1\}^{|B|}$. This vector is generated in the pricing problem and appended to a list of the surgeon's columns. Variable θ_k^s only indicates, whether a column is selected, but does not keep track of content of the column. For this purpose, we need to create a parameter $a_{k,b}^s \in \{0,1\}$.

$$a_{k,b}^s = 1 \Leftrightarrow \text{block } b \text{ is assigned to surgeon } s \text{ in schedule } k \tag{6.2}$$

So the parameter $a_{k,b}^s$ can be thought of as en element of a three-dimensional entity, where the length of the first dimension s is fixed (constant number of surgeons), the length of the second dimension k can vary (number of columns for each surgeon changes) and the third dimension b is always of length $|B|$.

In addition, we need to introduce two more parameters for each pattern k of surgeon s, parameter $\Delta_s^k \in \mathbb{R}$ and $w_s^k \in \mathbb{R}$. They will be explained later.

6.1.1 Master Problem

In the master problem, the constraints regarding the individual columns are omitted and are left for the subproblem. We only keep the constraint referring to 5.2. Other constraints in the master problem concern the variable θ_s^k:

1. for each surgeon, the sum of θ_s^k over all columns must be greater than 1

2. every column must be non-negative

We also introduce a set K_s, which comprises all columns of surgeon s in the current node. The formulation of the master problem is following:

$$\min \quad \alpha(C - \sum_{s \in S} \sum_{k \in K_s} \theta_k^s \Delta_k^s) + \beta \sum_{s \in S} \sum_{k \in K_s} \theta_k^s w_k^s \tag{6.3}$$

$$\equiv \min \sum_{s \in S} \sum_{k \in K_s} \theta_k^s (\beta w_k^s - \alpha \Delta_k^s) \tag{6.4}$$

$$s.t.$$

$$\forall d \in W, \forall t \in T: \quad \sum_{b \in O_{dt}} \sum_{s \in S} \sum_{k \in K_s} \theta_k^s a_{k,b}^s \leq |R| \tag{6.5}$$

$$\forall s \in S: \quad \sum_{k \in K_s} \theta_k^s \geq 1 \tag{6.6}$$

$$\forall s \in S, \forall k \in K_s: \quad \theta_k^s \geq 0 \tag{6.7}$$

6.1.2 Dual Master Problem

We convert the primal master problem to dual master problem by doing several steps. First, if primal is a minimization problem, the dual becomes a maximization problem. Second, every constraint in the primal becomes a variable in the dual. Also every variable in the primal becomes a constraint in the dual. In linear programming, we talk about strong duality. That means that an optimal solution for the primal problem is also an optimal solution for the dual problem.

In the dual master problem, two variables λ_{dt} and μ_s arise from the primal problem from constraints 6.5 and 6.6 respectively.

$$\max \quad \sum_{d \in W} \sum_{t \in T} |R| \lambda_{dt} + \sum_{s \in S} \mu_s \tag{6.8}$$

$$s.t.$$

$$\forall s \in S, \forall k \in K_s: \quad \sum_{d \in W} \sum_{t \in T} \sum_{b \in O_{dt}} a_{k,b}^s \lambda_{dt} + \mu_s \leq \beta w_k^s - \alpha \Delta_k^s \tag{6.9}$$

$$\forall d \in W, \forall s \in S, \forall t \in T: \quad \lambda_{dt} \leq 0 \tag{6.10}$$

$$\forall s \in S: \quad \mu_s \geq 0 \tag{6.11}$$

6.1.3 Subproblem

The subproblem, also called the pricing problem, is used to generate new columns. Unlike in the ILP formulation, the subproblem is solved for each surgeon separately. In each iteration of column generation, up to $|S|$ columns can be added, one column per each surgeon. In experiments, we also run the branch-and-price with only one column per iteration meaning that once a column is added, the subproblem stops and master problem is reoptimized again.

To represent a column that may enter the basis, we need to create a vector $\mathbf{o} \in \{0,1\}^{|B|}$. For this purpose, we define a new variable o_b.

$$o_b = 1 \Leftrightarrow \text{block } b \text{ is assigned to the surgeon in the current schedule} \tag{6.12}$$

In the subproblem, we also use variable x_{pb}, where $p \in P_s$, because the pricing problem is solved separately for each surgeon. For the same reason, we introduce a modified variable q_{lk} (we omit index s).

$$q_{lk} = 1 \Leftrightarrow \text{surgeon has exactly } k \text{ blocks of length exactly } l \tag{6.13}$$

The absence of variable y_{sb} leads to the following modifications of constraints 5.5 and 5.6 compared to the ILP model.

$$\forall l \in L: \quad \sum_{k=0}^{|W|} k q_{lk} = \sum_{b \in B:\ D_b = l} o_b \tag{6.14}$$

$$\forall l \in L: \quad \sum_{k=0}^{|W|} q_{lk} = 1 \tag{6.15}$$

A column is added to the restricted master problem, if it has a negative reduced cost. In order to compute the reduced cost, we need to retrieve the value of dual variables λ_{dt} and μ_s or corresponding primal constraints (thanks to strong duality). A schedule for surgeon s is added, if the following condition is fulfilled:

$$\sum_{d \in W} \sum_{t \in T} \sum_{b \in O_{dt}} o_b \lambda_{dt} + \mu_s - \beta w_s + \alpha \Delta_s > 0 \tag{6.16}$$

Expressions Δ_s and w_s are explained in the formulation of the subproblem, which is below.

for $\forall s \in S$ solve:

$$\max \quad \sum_{d\in W}\sum_{t\in T}\sum_{b\in O_{dt}} o_b\lambda_{dt} - \beta w_s + \alpha\Delta_s \tag{6.17}$$

$s.t.$

$$\sum_{b\in B} o_b \geq m \tag{6.18}$$

$$\forall d \in W: \quad \sum_{b\in B_d} o_b \leq 1 \tag{6.19}$$

$$\forall l \in L: \quad \sum_{k=0}^{|W|} kq_{lk} = \sum_{b\in B:\ D_b=l} o_b \tag{6.20}$$

$$\forall l \in L: \quad \sum_{k=0}^{|W|} q_{lk} = 1 \tag{6.21}$$

$$\forall p \in P_s: \quad \sum_{b\in B} x_{pb} \leq 1 \tag{6.22}$$

$$\forall b \in B: \quad \sum_{p\in P_s} \delta_p x_{pb} \leq D_b o_b \tag{6.23}$$

$$w_s = \sum_{p\in P_s} pr_p^{LP}(1 - \sum_{b\in B} x_{pb}) \tag{6.24}$$

$$\Delta_s = \sum_{p\in P_s}\sum_{b\in B} \delta_p x_{pb} \tag{6.25}$$

Follower's Problem

$$\max \quad f_s^{optim} \tag{6.26}$$

$s.t.$

$$\forall p \in P_s: \quad \sum_{b\in BA_s} x'_{pb} \leq 1 \tag{6.27}$$

$$\forall b \in BA_s: \quad \sum_{p\in P_s} \delta_p x'_{pb} \leq D_b \tag{6.28}$$

$$f_s^{optim} = \sum_{p\in P_s} \pi_p \sum_{b\in BA_s} x'_{pb} \tag{6.29}$$

$$w'_s = \sum_{p\in P_s} pr_p^{LP}(1 - \sum_{b\in BA_s} x'_{pb}) \tag{6.30}$$

$$\Delta'_s = \sum_{p\in P_s}\sum_{b\in BA_s} \delta_p x'_{pb} \tag{6.31}$$

The set BA_s of blocks assigned to surgeon s is defined in a similar way as in ILP model.

$$BA_s = \{b \in B : o_b = 1\} \tag{6.32}$$

6.2 Lazy Constraints

We only want to add such columns that are optimal also from the follower's perspective. This will once again be achieved by adding lazy constraints to the subproblem. The process of adding lazy constraints is similar as in section 5.2. Following changes are made:

- $o_b^{(c)}$ is the c-th block allocation found by the subproblem
- $n_{sl}^{(c)} = \sum_{b \in BA_s : D_b = l} o_b^{(c)}$ the number of blocks of duration l assigned to surgeon s related to solution c

6.2.1 Objective Value Based Lazy Constraint

The form of first lazy constraint is almost identical to 5.15. If $f_s < f_s^{optim}$ the following lazy constraint is added (value of f_s is computed in the same way as 5.14):

$$\sum_{p \in P_s} \pi_p \sum_{b \in B} x_{pb} + M \cdot \left(|L| - \sum_{l \in L} q_{l n_{sl}^{(c)}} \right) \geq f_s^{optim} \tag{6.33}$$

6.2.2 Assigned Patients Based Lazy Constraint

In the second lazy constraint, we slightly alter the objective function of the follower to

$$\max\ f_s^{optim} + \frac{1}{M}(\alpha \Delta_s' - \beta w_s')$$

The idea is to prioritize a patient-block assignment with better leader's objective in case more assignments have the same f_s^{optim} value. In this case, the big M is usually a large value e.g. $M = 1000$.

We obtain the set of patients assigned PA_s in the subproblem the same way as in 5.16. Then, if $f_s < f_s^{optim}$, we add the following constraint:

$$\sum_{p \in PA_s} \sum_{b \in B} x_{pb} - \sum_{p \in P_s \setminus PA_s} \sum_{b \in B} x_{pb} + M \cdot \left(|L| - \sum_{l \in L} q_{l n_{sl}^{(c)}} \right) \geq |PA_s| \tag{6.34}$$

6.3 Initial Columns

In beginning of the branch-and-price algorithm, we require an initial set of columns. The most easiest way to generate an initial column for each surgeon, which would be bilevel feasible, is to create an empty column. Such column would be represented by a zero vector of size $|B| \times 1$. So at the start, a three-dimensional array $\mathbf{a}$ with parameter $a^s_{k,b}$ as an element would look like:

$$\mathbf{a} = \begin{bmatrix} \begin{array}{c} s_0 \\ \begin{bmatrix} \begin{array}{c} k_0 \\ \begin{bmatrix} 0\\0\\0\\\vdots\\0\\0\\0 \end{bmatrix} \end{array} \end{bmatrix} \end{array}, \begin{array}{c} s_1 \\ \begin{bmatrix} \begin{array}{c} k_0 \\ \begin{bmatrix} 0\\0\\0\\\vdots\\0\\0\\0 \end{bmatrix} \end{array} \end{bmatrix} \end{array}, \dots \begin{array}{c} s_{|S|-2} \\ \begin{bmatrix} \begin{array}{c} k_0 \\ \begin{bmatrix} 0\\0\\0\\\vdots\\0\\0\\0 \end{bmatrix} \end{array} \end{bmatrix} \end{array}, \begin{array}{c} s_{|S|-1} \\ \begin{bmatrix} \begin{array}{c} k_0 \\ \begin{bmatrix} 0\\0\\0\\\vdots\\0\\0\\0 \end{bmatrix} \end{array} \end{bmatrix} \end{array} \end{bmatrix}$$

For every column k of surgeon s, we need to store values of Δ^k_s and w^k_s. For empty patterns (denoted as k_0), surgeons cannot schedule any patients as they have no blocks assigned. It means that variable x_{pb} will also be zero for every (p, b) pair. Because of that, it's simple to compute parameters $\Delta^{k_0}_s$ and $w^{k_0}_s$.

$$\Delta^{k_0}_s = \sum_{p \in P_s} \sum_{b \in B} \delta_p x_{pb} = 0 \tag{6.35}$$

$$w^{k_0}_s = \sum_{p \in P_s} pr^{LP}_p (1 - \sum_{b \in B} x_{pb}) = \sum_{p \in P_s} pr^{LP}_p \tag{6.36}$$

6.3.1 Initial Heuristics

It this section, we present an initial heuristics, which creates non-trivial initial columns for each surgeon. It is based on iterative adding of surgeon-block assignments/pairs to the schedule (Maenhout et al., 2023). In the following subsections, we will introduce the algorithm and we will also define several auxiliary models.

Knapsack Model

In each iteration, we try to solve a knapsack problem with objective 6.37 and constraint 6.38. The knapsack is solved for every surgeon $s \in S$ and every block length $l \in L$. A set P'_s represents remaining patients of surgeon s who have not yet been assigned. Variable x_p signify that patient p is assigned.

$$\min \quad \alpha(l - \sum_{p \in P'_s} \delta_p x_p) + \beta \sum_{p \in P'_s} pr_p^{LP}(1 - x_p) \tag{6.37}$$

$$s.t.$$

$$\sum_{p \in P'_s} \delta_p x_p \leq l \tag{6.38}$$

When the model is solved, we store two values corresponding to the (s, l) pair: idle capacity $\bar{\Delta}_{sl} = (l - \sum_{p \in P'_s} \delta_p x_p)$ and utilization function $\bar{u}_{sl} = \sum_{p \in P'_s} pr_p^{LP} x_p$.

Surgeon-Block Assignment Model

This model assigns blocks to surgeons via variable y_{sb}. The surgeon-block assignments created in the previous iteration are stored in a set ϕ, which is empty in the beginning of the algorithm. It also uses values from the knapsack problem $\bar{\Delta}_{sl}$ and $\bar{u}_{sl}$. The objective is similar to the leader's objective 5.1. We try to minimize the idle time and the priority penalty associated with patients that are not yet included in the schedule. In addition, we prioritize blocks, which start earlier. For this reason, we define parameter T_b for each block as follows:

$$T_b = d_b \times (|T| + 1) + t_b, \tag{6.39}$$

where d_b and t_b are day and start time of a block respectively.

Constraint 6.41 stipulates that the surgeon-block assignments made earlier cannot be altered. We might require at least θ blocks to be scheduled in each iteration (constraint 6.46). Furthermore, constraint 6.43 assures that at most one block can be scheduled for surgeon s per iteration. Contraints 6.42, 6.44 and 6.45 come straight from the ILP model.

$$\min \quad \sum_{s \in S} \sum_{b \in B} (\alpha \bar{\Delta}_{sD_b} - \beta \bar{w}'_{sD_b} + T_b) y_{sb} \tag{6.40}$$

$$s.t.$$

$$\forall (s, b) \in \phi : \quad y_{sb} = 1 \tag{6.41}$$

$$\forall s \in S, \forall d \in W : \quad \sum_{b \in B_d} y_{sb} \leq 1 \tag{6.42}$$

$$\forall s \in S : \quad \sum_{b \in B \setminus \phi} y_{sb} \leq 1 \tag{6.43}$$

$$\forall d \in W, \ \forall t \in T : \quad \sum_{s \in S} \sum_{b \in O_{dt}} y_{sb} \leq |R| \tag{6.44}$$

$$\forall s \in S : \quad \sum_{b \in B} y_{sb} \geq m \tag{6.45}$$

$$\sum_{s \in S} \sum_{b \in B \setminus \phi} y_{sb} \geq \theta \tag{6.46}$$

$$\forall s \in S, \forall b \in B : \quad y_{sb} \in \{0, 1\} \tag{6.47}$$

Algorithm

Below is a pseudocode for initial heuristics algorithm.

Algorithm 2 Initial heuristic

```
    *Step 0: Initialization*
 1: $\phi = \emptyset$                    ▷ Block schedule - assigned surgeon-block pairs
 2: **for** $s \in S$ **do**
 3:     $P'_s = P_s$
 4: **repeat**
    *Step 1: Compose best block composition for every block duration*
 5:     **for** $s \in S : P'_s \neq \emptyset$ **do**
 6:         **for** $l \in L$ **do**
 7:             Solve knapsack problem (6.37)-(6.38) and store values of $\bar{\Delta}_{sl}$
                and $\bar{u}_{sl}$
    *Step 2: Schedule one additional block per surgeon to block schedule*
 8:     Solve model (6.40)-(6.47) with current block schedule, $\bar{\Delta}_{sl}$ and $\bar{u}_{sl}$ as
        input
    *Step 3: Update data*
 9:     Update current block schedule $\phi$ with newly assigned blocks
10:     **for** $s \in S$ **do**
11:         Update $P'_s$ by removing patients that have been assigned in this
            iteration
12: **until** Model (6.40)-(6.47) returns a feasible solution (= new surgeon-block
    pair is added to the block schedule)
    *Step 4: Find bilevel feasible patient schedule based on selected blocks*
13: **for** $s \in S$ **do**
14:     Solve follower's problem model (5.10)-(5.13) given currently assigned
        blocks $y_{sb}$ resulting from the current block schedule
    *Step 5: Evaluation*
15: Evaluate patient schedule obtained in Step 4 according to leader's objec-
    tive 5.1 (will be set as incumbent solution of the branch-and-price)
```

6.4 Lazy Constraint Remembering

One of the speed-up mechanisms proposed in (Maenhout et al., 2023) is lazy constraint remembering. There is an assumption that same lazy constraints are generated multiple times. For the sake of effectiveness, we can save the lazy constraints generated for surgeon s to a set LC_s and then add them explicitly to the subproblem in other iterations. It is expected that there will be lower number of generated lazy constraints, which will result in lower time spent in the subproblem. Impact of this speed-up mechanism is shown in the section 8.

6.5 Branching

Suitable branching strategy is important for computational performance of the branch-and-price algorithm. One option is to branch on column variables. In (Vanderbeck, 2000), authors suggest that such approach can lead to unbalanced branch-and-bound trees and can cause significant modifications to the subproblem. More convenient method is branching on original variables from the ILP formulation. Authors in (Villeneuve et al., 2005) show that such a formulation always exists under certain assumptions. In our case, we branch over the variable y_{sb}. The branching separates surgeon's patterns into two disjunctive sets: first set are all columns satisfying $y_{sb} = 0$ and the other are all columns with $y_{sb} = 1$. In every node, we keep track of the branching history from the root node down to the current node that we refer to as BH.

$$(s, b, v) \in BH, \quad \text{meaning } y_{sb} = v \tag{6.48}$$

The branching decisions are added to the pricing problem as constraints, assuring consistency and correctness.

Algorithm 3 Adding constraints from branching history to the subproblem

```
for $(s, b, v) \in BH$ do
    if $s$ is the surgeon who we solve the subproblem for then
        Add constraint $o_b = v$ to the subproblem model
```

When the column generation is terminated and the solution is fractional, we must decide, what (s, b) pair we are going to branch over. The "most fractional" variable is usually chosen. The most fractional variable is the variable with the closest value to 0.5 (in binary case). Also a "worst assignment" can be used (Maenhout and Vanhoucke, 2009). In that case, a fractional variable that is worst in terms of objective function value is selected.

As we don't explicitly have the y_{sb} variable, we need to compute it using master problem variable θ_s^k and a vector $\mathbf{a}_k^s$, which represents column k of surgeon s. The most fractional variable is obtained as follows:

$$(s_{br}, b_{br}) = \underset{s \in S, b \in B}{\operatorname{argmin}} \left(\sum_{k=0}^{|K|-1} \mathbf{a}_k^s \theta_k^s - 0.5 \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix} \right) \tag{6.49}$$

After we obtain the branching variable, we split the columns into two sets according to the new branching decision. Then, we branch with $y_{s_{br}b_{br}} = 0$ and $y_{s_{br}b_{br}} = 1$ and we add (s_{br}, b_{br}, v) to BH accordingly.

An example of a search tree in branch-and-price is shown in figure 6.1. It can be seen that an incumbent solution was found at depth 3 and because lower bound of all the parents is equal to the incumbent solution, there is no need to explore other nodes.

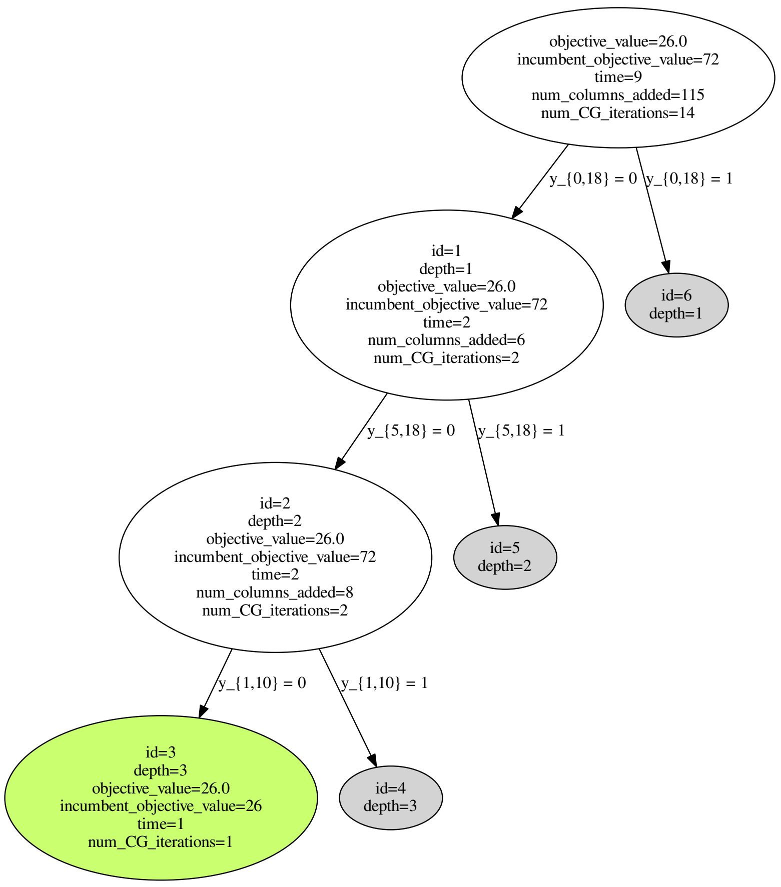


Figure 6.1: Search tree of a branch-and-price.

6.6 Dummy Pattern

In the beginning of branch-and-price algorithm, we have an initial set of columns for each surgeon (empty pattern + pattern from initial heuristics). During branching, whenever a block is assigned to a surgeon, the empty pattern disappears. Thus, a following situation can occur:

- surgeon 0 has only one column in which he is assigned block 0
- surgeon 1 has only one column in which he is assigned block 1
- block 0 and block 1 overlap
- there is only one operating room $|R| = 1$

It's obvious that it is impossible to solve the master problem in this case. By constraint 6.6 we are obliged to assign a schedule for every surgeon, but constraint 6.5 enforces that at most one of surgeon-block assignments above is assigned as they are overlapping and only one operating room is available.

To tackle this issue, we need to generate a "dummy" pattern every time we branch with value $v = 1$, i.e. a block is firmly assigned to a surgeon. The pattern is generated based on branching history. The procedure of generating a "dummy" pattern is shown in the pseudocode below.

Algorithm 4 Dummy pattern generation

```
Require: $s_{br}, b_{br}$                ▷ Branching pair (already added to $BH$)
1: $dummyPattern \leftarrow \emptyset$
2: for $(s, b, v) \in BH$ do
3:     if $s_{br} = s$ then
4:         Add block $b$ to $dummyPattern$
5: if $dummyPattern$ in $K_{s_{br}}$ then          ▷ $K_{s_{br}}$ - patterns of surgeon $s$
6:     quit
7: else
8:     Solve follower's problem (5.10)-(5.13) and retrieve parameters $\Delta_{dummy}$
       and $w_{dummy}$
9:     Add $dummyPattern$ to $K_{s_{br}}$ and alongside save parameters $\Delta_{dummy}$
       and $w_{dummy}$
```

Chapter 7

Data Analysis

In this section, we analyze two different datasets. First is a publicly available dataset called CHOIR that was published in paper (Leeftink and Hans, 2017). The second dataset are real-life data from University Hospital of Hradec Králové.

7.1 CHOIR Dataset

The name comes from *Centre for Healthcare Operations Improvement and Research* at University of Twente in Netherlands. The dataset is generated based on a characterization of the patient cases and surgery types. The full dataset contains 22,400 instances (ten diverse instances per parameter combination). A smaller test set containing only 146 instances is also present in the data set. The instances are generated based on both theoretical and real-life case mixes, encompassing 11 surgical specializations.

Each instance is characterized by the number of OR days, load and overall capacity of operating rooms in minutes. Then, the instance contains a list of surgical cases. Each surgical case is represented by a distribution type and its parameters. In literature, researchers mostly use a 3-parameter lognormal distribution to describe the duration δ_p of a surgery (Stepaniak et al., 2009)(May et al., 2000). The probability density function of a 3-parameter log-normal distribution is defined as follows:

$$f(x) = \frac{1}{(x-\gamma)\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x-\gamma)-\mu)^2}{2\sigma^2}} \tag{7.1}$$

The same distribution with parameters is also used in the CHOIR dataset. As stated in (Leeftink and Hans, 2017), the three parameters are $\mu \in (0, \infty)$, $\sigma \in (0, \infty)$ and $\gamma \in [0, \infty)$. From these parameters, an average duration m and standard deviation s can be computed as:

$$m = \gamma + e^{\mu + \frac{\sigma^2}{2}} \tag{7.2}$$

$$s = \sqrt{(e^{\sigma^2} - 1) \times e^{2\mu+\sigma^2}} \tag{7.3}$$

However, when fitting data onto a log-normal distribution in *SciPy* library in Python, we get three parameters *shape*, *location*, *scale*. The relation between these three parameters and the parameters μ, σ, γ is following:

$$\mu = \ln(scale) \tag{7.4}$$
$$\sigma = shape \tag{7.5}$$
$$\gamma = location \tag{7.6}$$

In the dataset, the authors use load ranging from 0.8 to 1.2. To induce selection of patients (not assigning all of them), we do not consider instances with load 1 or smaller. From the available instances, we present a brief summary of the data.

	Surgery duration (in minutes)		
Load	m	Range	s
1.05	105	[15,375]	73.50
1.10	105	[15,420]	71.40
1.15	101	[15,450]	73.73
1.20	99	[15,420]	71.28

Table 7.1: Instance characteristics averaged per load

When looking at the data, it is obvious that many parameters required for our problem are missing. First, the individual surgical cases are not related to any particular surgeon. We do it arbitrarily by choosing a random surgeon $s \in S$ (with discrete uniform distribution). The patient's priorities are also not present, so we generate both the leader's and the follower's priorities from a discrete uniform distribution on interval $[1, 3]$.

7.2 Data from University Hospital of Hradec Králové

The data are records of real life surgical cases that took place between years 2018 and 2019. Unlike the CHOIR dataset, we have information about the deterministic values of the surgery duration and the pre-estimated duration. The surgery duration is measured as the time difference between the patient's arrival and departure from the OR. We also have information about which surgeon is responsible for each patient.

First, the actual probability distribution of the data (surgery duration) must be verified. As stated earlier, the literature mostly assumes lognormal distribution (and so does the CHOIR dataset). Some papers also work with gamma distribution (Choi and Wilhelm, 2012). We will try to fit our data onto the most common distributions (including lognormal and gamma). For this purpose, we will assume four distinct surgical specializations - urology, trauma surgery, general surgery, robotic surgery. For each of these specializations, we will do the following steps:

1. select all surgical cases that belong to the specialization

2. try to fit the data onto the most common distributions (Cauchy, chi-squared, exponential, exponential power, gamma, lognormal, normal, power law, Rayleigh, uniform)[1]

3. retrieve the best probability distribution, lognormal distribution and gamma distribution and their parameters

4. draw samples from these three distributions and perform a Kolmogorov–Smirnov test for each of them[2]

Below is a table, where the best probability distribution is displayed for each specialization along with the p-value for the best, lognormal and gamma distribution.

Specialization	Best distr.	p-value	Lognormal's p-value	Gamma's p-value
Urology	gamma	0.35	0.33	0.35
Trauma surgery	gamma	0.68	0.5e−09	0.68
General surgery	lognormal	0.33	0.33	0.26
Robotic surgery	Cauchy	0.07	1.81e−04	1.97e−04

Table 7.2: Distributions' p-values for different specializations

As expected, the data indicates that lognormal and gamma distributions indeed are suitable for describing surgery duration as a random variable. In this case, the gamma distribution shows a bit more promising results, because according to Kolmogorov-Smirnov test, it fits 3 out of 4 specializations (lognormal only 2). The only anomaly is the robotic surgery that suggests the data follows Cauchy distribution.

We can also compare the parameters such as mean and standard deviation as in table 7.1. Instead of dividing the surgical cases by the load, in real-life data we split them according to the specialization.

	Surgery duration (in minutes)		
Specialization	m_t	Range	s_t
Urology	124	[20,430]	66.96
Trauma surgery	163	[30,700]	109.38
General surgery	127	[30,515]	67.58
Robotic surgery	220	[32,645]	85.79

Table 7.3: Real-life data characteristics per specialization

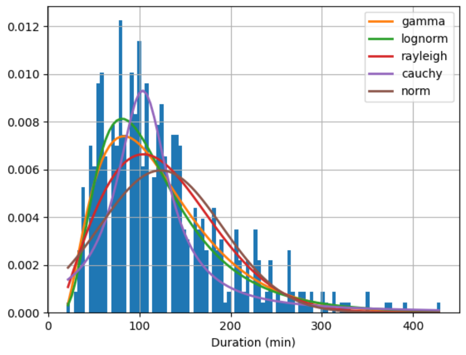
The standard operating time in ORs in University Hospital of Hradec Králové is usually 8 hours (from 7 AM to 3 PM). Thus, patients with duration longer than 8 hours (480 minutes) will never fit into any block, so they will stay unscheduled.

Distributions of the four specializations are also plotted below, together with probability distribution functions for the TOP 5 distributions. It can be

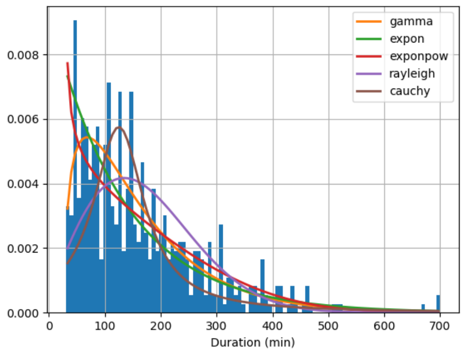
[1]We use Fitter Python library to fit the distributions

[2]The Kolmogorov-Smirnov test is a method of mathematical statistics that serves to test whether two univariate random variables come from the same probability distribution

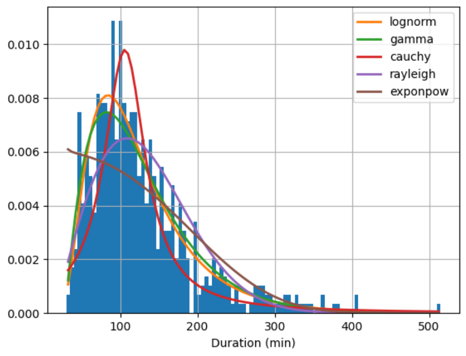
seen that for urology, trauma surgery and general surgery, the distributions are skewed and so they fit log-normal/gamma distribution decently. In case of robotic surgery, there are not many short surgeries, so the probability distribution functions are not so skewed.



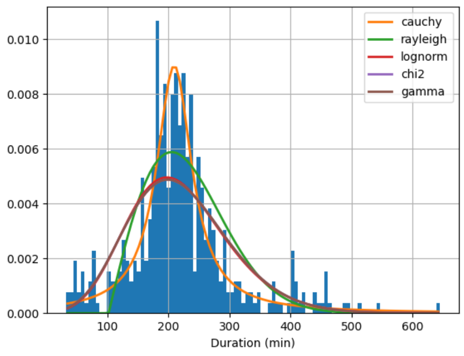
(a) Urology



(b) Trauma surgery



(c) General surgery



(d) Robotic surgery

Figure 7.1: Comparison of TOP 5 distributions for different specializations

Chapter 8
Experimental Results

In this chapter, we present the experiments carried out on our ILP and branch-and-price models. In section 8.1, we describe the software and hardware used to implement the algorithms. The logical structure of the code is also explained. In the next section 8.2, we introduce several experiments, where we compare the performance of ILP and branch-and-price model, the solution quality and speed-up mechanisms.

8.1 Implementation

The whole implementation is written in Python. This includes loading data, instance generation, ILP and branch-and-price algorithm. The solver used to solve MILP problems is *Gurobi* (the Python interface is called *gurobipy*). Just as in CHOIR dataset, we use text format to store instances of our real data. The parameters are stored in JSON format.

The scripts can be run on any machine with Python and Gurobi, but for the sake of computational performance, we use Slurm cluster for experiments. Slurm is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters. On our cluster we can make use of up to 6 nodes, each having either 256 or 512 GB of RAM and two Intel Xeon E5-2690 v4 processors @ 2.60GHz.

Below is a scheme of the project structure. In the *configs* folder, we have two subfolders - *instances* and *parameters*. All the outputs are saved into a folder of the same name. Results from benchmarks are stored into folder *excel*. To store graphical outputs, we have folders *schedules* that serve to save timeline charts and *trees*, where branch-and-price trees are saved. Gurobi models are saved to *models* folder. In order to run experiments on our cluster, we have created multiple bash scripts in *scripts* folder. Python files with ILP and branch-and-price algorithm implementation are in *src* folder. Subfolder *benchmarks* have multiple Python scripts to run benchmark experiments.

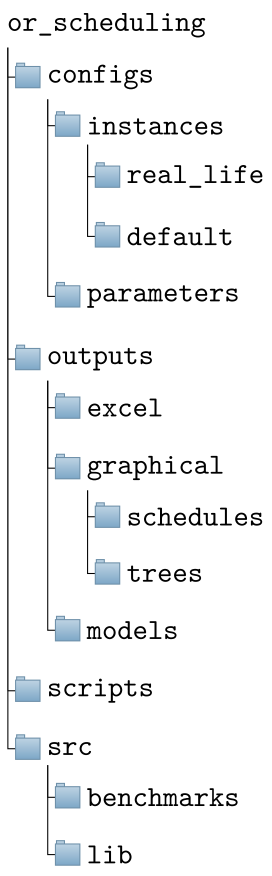


Figure 8.1: Project's folder structure

8.2 Computational Results

8.2.1 Comparison of Branch-and-Price and ILP Model

In the first set of experiments, we compare the performance of the baseline ILP model and the branch-and-price. We put emphasis on overall time and percentage of instances solved to optimality. We use instances that are created from real-life data. For simplicity, we will refer to objective value based lazy constraint as $LC1$ and the assigned patients based lazy constraint as $LC2$. We will consider a timeout of 1800 seconds. When an instance is not solved within this time, it will be flagged as not optimal, even though the objective might have optimal value.

Small Instances

By small instances, we mean instances with relatively small number of surgeons, patients and operating rooms. For this purpose, we create instances with the following specifications. We set number of operating rooms $|R| = 1$. Then, each instance has number of surgeons $|S|$ from interval $[13, 22]$. Number of patients $|P|$ for each instance ranges on interval $[41, 60]$. We use all off the speed-up techniques - initial heuristics and LC remembering. The type of lazy constraint is $LC2$. We consider 5 days as planning time horizon.

	Integer linear program			**Branch-and-price**		
Instance	Time [s]	Optimal	Objective	Time [s]	Optimal	Objective
january_|R|=1	915.3	True	57	65.4	True	57
february_|R|=1	701.0	True	56	49.5	True	56
march_|R|=1	1800.0	False	50	22.9	True	50
april_|R|=1	42.0	True	57	58.6	True	57
may_|R|=1	1800.0	False	59	50.2	True	59
june_|R|=1	8.5	True	51	19.8	True	51
july_|R|=1	46.9	True	25	14.4	True	25
august_|R|=1	5.8	True	46	75.8	True	46
september_|R|=1	15.7	True	46	23.9	True	46
october_|R|=1	107.6	True	39	17.2	True	39
november_|R|=1	9.3	True	39	128.6	True	39
december_|R|=1	0.2	True	39	8.1	True	39
average	454.37	0.83	47	44.52	1.00	47

Table 8.1: Benchmark comparison for small instances

From the table above, we can see that for all the instances, ILP and branch-and-price model yields the same integer objective. The average time for the branch-and-price is significantly better compared to the ILP, although it is caused by the two time-outed instances. There is 83 % of instances solved to optimality for the ILP and 100 % for the branch-and-price.

Large Instances

The drawbacks of ILP model in terms of efficiency will arise when we make instances larger. We could for example set number of operating rooms to $|R| = 4$. Every instance has number of surgeons $|S|$ from interval $[43, 62]$. Number of patients $|P|$ for each instance ranges on interval $[162, 206]$. Once again, all speed-up techniques are used and the lazy constraint type is $LC2$. We consider $|D| = 5$.

	Integer linear program			**Branch-and-price**		
Instance	Time [s]	Optimal	Objective	Time [s]	Optimal	Objective
january_|R|=4	1800.0	False	178	88.2	True	178
february_|R|=4	1800.1	False	148	46.8	True	148
march_|R|=4	1800.0	False	179	67.2	True	179
april_|R|=4	1800.0	False	172	269.7	True	172
may_|R|=4	1800.0	False	172	60.8	True	172
june_|R|=4	1800.0	False	159	47.6	True	159
july_|R|=4	1800.0	False	119	38.7	True	119
august_|R|=4	1800.1	False	125	59.9	True	125
september_|R|=4	1800.1	False	163	180.3	True	163
october_|R|=4	1800.0	False	152	41.2	True	152
november_|R|=4	1800.0	False	170	65.4	True	170
december_|R|=4	1800.0	False	88	16.8	True	88
average	1800.0	0.00	152.08	81.94	1.00	152.08

Table 8.2: Benchmark comparison for large instances

From the table above, it is obvious that ILP model becomes incompetent for solving large instances. Similar results are obtained when working with the CHOIR dataset, where ILP has turned out to be more efficient on smaller instances than branch-and-price, but inefficient for larger instances.

Gantt Charts

Below is an example of two Gantt charts corresponding to the ILP and branch-and-price results, where surgeon-block assignment and patient-block assignment are shown - first with one operating room, second with four operating rooms.

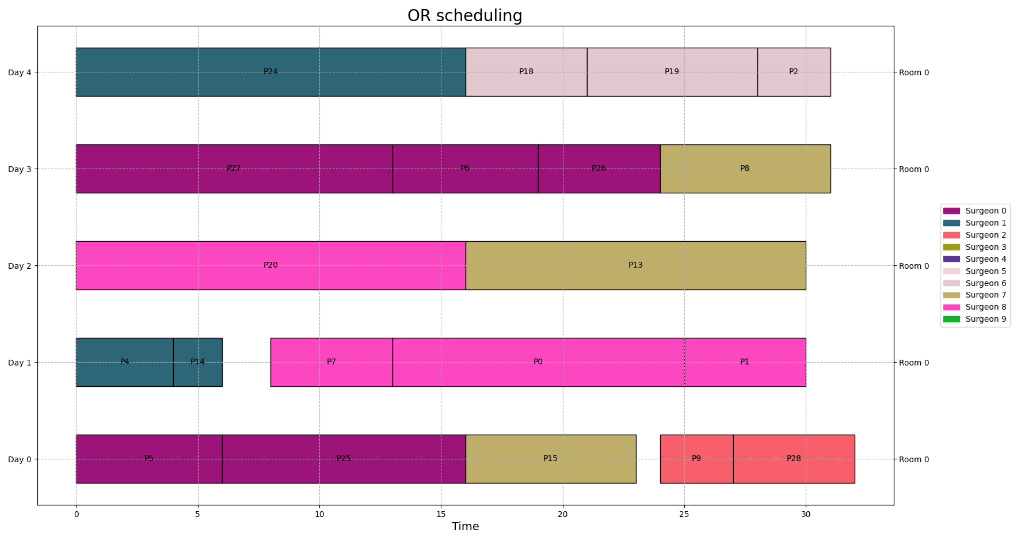


Figure 8.2: Gantt chart for schedule with one operating room.

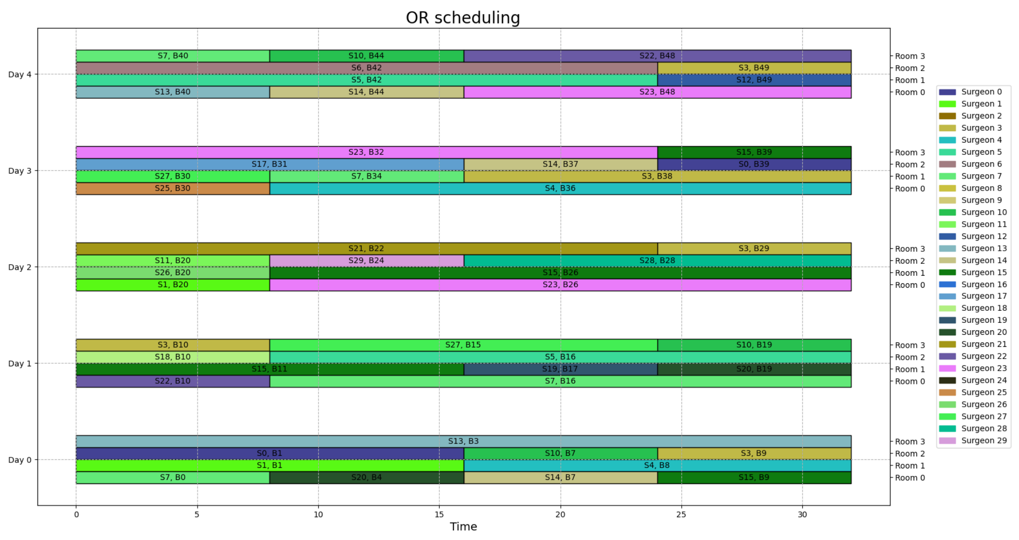


Figure 8.3: Gantt chart for schedule with four operating rooms.

From figure 8.3, we can see that the head of surgeons always fills in the entire available capacity. On the other hand, the assignment of patients may contain many time gaps (idle time). This can be seen in figure 8.2. Time gaps, of course, are not in contradiction with the bilevel optimality of the schedule.

Comparing Different Configurations

During testing, one can create many configurations. In the following experiment, we compare the performance and quality of different tested problem sizes and different types of lazy constraints. The solution quality is expressed by a value F, which is the value of leader's objective value, the value of the final lower bound F^{LPR} that relaxes the binary domain conditions of both the leader and follower decision variables, the relative optimality gap $\%Gap = \frac{F - F^{LPR}}{F}$ and the percentage of instances solved to optimality $\%Opt$. Once again, we will measure the computational performance by run time in seconds. We compute the lower bound F^{LPR} for the branch-and-price as the relaxed objective in the root node.

We will create the following configurations.

- $|S| \in \{10, 15\}$ if $|R| = 1$
- $|S| \in \{30, 45\}$ if $|R| = 4$
- $|D| \in \{5, 10\}$ time horizon of 1 or 2 weeks
- lazy constraints $= \{LC1, LC2\}$

Together, there will be 16 configurations. This will require 16 instances to be created. They are compared in table 1 that can be found in the appendix.

8.2.2 Impact of Speed-Up Mechanisms

In this section, we will focus on evaluating the impact of the three proposed speed-up mechanisms for the branch-and-price model.

1. Initial heuristics (InH)
2. Generating multiple patterns per pricing iteration (MuP)
3. LC remembering (LCR)

The idea is to start with a version, where all speed-up mechanisms are present. Then, we leave out each of the three accelerating technique and the rest of algorithm stays the same. Altogether, there will be four versions for every lazy constraint.

In this table, we report the value of column generation iteration ($\#CGIter$), the number of columns generated ($\#columns$), lazy constraints generated in the callbacks ($\#LC$), callbacks ($\#CB$), nodes in the search tree ($\#nodes$). We further measure the solution quality by the relative optimality gap ($\%Gap$) and the percentage of instances solved to optimality ($\%Opt$). We also measure computational performance, i.e., the run time spent in the callback function to generate lazy constraints ($Time^{CB}$) and run time to solve the subproblem (including the callback time) ($Time^{SP}$), the run time to solve the master problem ($Time^{MP}$) and the total required run time ($Time^{T}$).

Furthermore, the values in this table are an average over multiple instances. The experiment was run for both the real dataset from Universital Hospity of Hradec Králové and the CHOIR dataset. The results are presented below.

Results From CHOIR Dataset

	LC1				**LC2**			
Mechanism	w/o InH	w/o MuP	w/o LCR	All	w/o InH	w/o MuP	w/o LCR	All
$\#CGIter$	236.1	480.4	258.5	243.8	171.1	223.5	252.4	74
$\#columns$	426.7	428.9	464.4	446.9	332.9	213.9	450.8	215.5
$\#LC$	4.3	4.2	772.8	4.4	4.2	4.5	767	4.8
$\#CB$	5750.8	5258.9	7053.5	6139.8	3776.2	1721	6902.7	1723.1
$\#nodes$	92.4	54.2	101	98.2	44.2	9.6	99	21.6
$\%Gap$	0.02	0.02	0.03	0.02	0.02	0.01	0.03	0.01
$\%Opt$	0.9	0.9	0.9	0.9	1	1	0.9	1
$Time^{CB}$ [s]	5.07	4.8	6.97	5.51	3.59	1.78	6.93	1.75
$Time^{SP}$ [s]	44.56	10.38	50.36	46.75	32.92	2.22	49.44	15.45
$Time^{MP}$ [s]	158.03	215.42	158.75	165.4	129.9	133.23	161.31	52.39
$Time^{T}$ [s]	206.54	259.35	211.02	212.58	163.07	148.94	212.55	67.97

Table 8.3: Comparison of speed-up mechanisms on the CHOIR dataset.

From the table above, it appears that using the advanced lazy constraint (LC2) yields better overall results than using the basic one (LC1). Row $\#LC$

indicates that the number of lazy constraints generated when LC remembering is turned off is huge compared to other versions. As for the run time, the version with advanced lazy constraint and all speed-up mechanisms on seems to be the fastest.

Results From Real-life Dataset

	LC1				**LC2**			
Mechanism	w/o InH	w/o MuP	w/o LCR	All	w/o InH	w/o MuP	w/o LCR	All
#CGIter	528.15	691.5	626.1	543.7	429.35	914.15	641.55	450.7
#columns	1105.4	655.1	1218.9	1098.05	895.25	862.75	1311.9	930.6
#LC	29.2	26	2615.35	30.75	40.15	37.95	3103	43.7
#CB	7327.3	4513.8	10794.8	6947.45	5940.3	6221.5	11979.1	6094.1
#nodes	125.4	46.7	128.9	126.3	79.2	57.5	144.2	88
%Gap	0.03	0.04	0.03	0.03	0.03	0.04	0.03	0.03
%Opt	0.65	0.7	0.65	0.65	0.85	0.75	0.6	0.9
$Time^{CB}$	13.42	8.78	25.17	13.27	13.41	14.51	35	13.48
$Time^{SP}$	285.18	34.69	214.01	259.89	162.16	26.42	243.11	170.1
$Time^{MP}$	298.73	409.22	351.07	319.81	257.65	565.85	359.36	267.7
$Time^{T}$	682.05	689.11	668.55	689.99	510.61	778.56	769.73	449

Table 8.4: Comparison of speed-up mechanisms on the real-life dataset.

From the results, the most preferable configuration in terms of run time is again the version with advanced lazy constraint and all speed-up techniques turned on. This configuration also has the biggest ratio of instances solved to optimality. Leaving out the LC remembering results in the large amount of lazy constraints generated and many callbacks. Also, using the advanced lazy constraints yields slightly better run times (with average 626.9 [s] over different versions) than the basic lazy constraints (average 682.4 [s]).

8.2.3 Game-Theoretical Approach

In this section, we evaluate the efficiency of the equilibrium solution obtained from our bilevel optimization problem. In the equilibrium solution, the individual surgeons (a.k.a. agents) cannot improve their objective given the allocated OR blocks, which are determined by the head of surgeons. To model the selfish behaviour of the surgeons, many concepts can be used e.g. Nash equilibrium. We will further present two ways to assess the quality of the Nash equilibrium.

- The *Price of Stability* (*PoS*) measures how the efficiency of a system degrades due to the required equilibrium (Paccagnan et al., 2022). It is a ratio between the best equilibrium solution and the best centralized solution. In our case, we evaluate the best equilibrium solution by solving the model (5.1)-(5.13). In other words, it is the bilevel optimal solution of our problem. The best centralized solution is the best solution of the leader's problem (5.1)-(5.8), so the followers' interest is not taken into consideration (in our implementation, we simply do not call the callback function). The price of stability for a minimization problem is computed by the following formula:

$$PoS = \frac{min_{e' \in E'} F(e')}{min_{e \in E} F(e)} \tag{8.1}$$

with E the set of all solutions and E' the set of equilibrium solutions.

- The *Price of Decentralization* (PoD) measures how the efficiency of a system degrades due to selfish behaviour of its agents. Similarly to the PoS, the Price of Decentralization is a ratio between the best decentralized solution and the best centralized solution. It models a situation, where surgeons plan their schedules without taking the leader's objective into consideration. We obtain the best decentralized solution by solving the model that maximizes the sum of the followers' objectives, i.e. the following objective

$$\max \quad \sum_{s \in S} f_s = \sum_{s \in S} \sum_{p \in P_s} \pi_p \sum_{b \in B} x_{pb} \equiv \sum_{p \in P} \pi_p \sum_{b \in B} x_{pb} \tag{8.2}$$

with respect to the constraints (5.2)-(5.8). The Price of Decentralization is formalised as

$$PoD = \frac{F(e \in E | max_{e \in E} \sum_{s \in S} f_s(e))}{min_{e \in E} F(e)}. \tag{8.3}$$

We decided to choose the metric of the Price of Decentralization over the Price of Anarchy (Paccagnan et al., 2022), which is defined as

$$PoA = \frac{max_{e' \in E'} F(e')}{min_{e \in E} F(e)}. \tag{8.4}$$

The Price of Anarchy is a ratio between the worst equilibrium solution and the best centralized solution. In our case, the worst equilibrium solution corresponds to an empty schedule, where the head of surgeons does not assign any blocks to any surgeons, thus the individual surgeons cannot schedule any patients. Such schedule is considered irrational.

For the next experiment, we are going to calculate the Price of Stability and the Price of Decentralization for different settings. Individual settings are defined by (i) patient's priorities from the leader's and the follower's perspective, (ii) the leader's priorities in the objective 5.1, defined by constants α and β. By default $\alpha = \beta = 1$, which makes the idle time and the penalty for not performing surgeries equally important. We will try to set one of these values to zero, eliminating the corresponding part of the objective. Furthermore, we will create 5 scenarios based on the patient's priorities:

- **Scenario 1** ($\pi_p = pr_p^{LP} = 1$): Patient's priorities are equal for all patients and at the same time, the leader's and the follower's priority are equal.
- **Scenario 2** ($\pi_p = pr_p^{LP} \in [1, 3]$ (random)): The leader's and the follower's priority of a patient are equal, but they can differ between patients.
- **Scenario 3** ($\pi_p = 1$; $pr_p^{LP} \in [1, 3]$ (random)): Patient's priorities set by the head of surgeons can be different between patients. These priorities may differ from the ones set by the individual surgeons, who set all patient priorities equal.
- **Scenario 4** ($\pi_p \in [1, 3]$ (random); $pr_p^{LP} = 1$): Patient's priorities set by the individual surgeons can be different between patients. For the head of surgeons, the patients have equal priority.
- **Scenario 5** ($\pi_p \in [1, 3]$ (random); $pr_p^{LP} \in [1, 3]$ (random)): Both patient's priorities are different between patients.

For the next table, we will need three models:

1. model that finds the best equilibrium solution, i.e. bilevel optimal solution BOM
2. model that finds the best decentralized solution DeM
3. model that finds the best centralized solution CeM

For each of the models, we will define the following metrics: (i) utilization of capacity U, (ii) F, which is the value of the objective function in 5.1, (iii) time T in seconds. Values in the table are averaged over multiple instances. The whole table 2 can be found in appendix.

Table 2 reveals that utilization of operating rooms is the highest for centralized solution, which corresponds with the objective of the leader, who unlike the surgeons, tries to avoid idle time within schedule. The worst utilization is expectedly for the decentralized solution. If we evaluate a model that seeks the worst equilibrium, the utilization would be zero as no patients would be in the schedule. The Price of Stability appears to be fairly small with an average of 1.05, which indicates that the equilibrium solution is close to the centralized solution. The Price of Decentralization has an average of 2.02 over the different configurations, so the price we would pay for selfish behaviour of the surgeons if we let them create the schedule is not negligible.

Chapter 9

Conclusion

The main objective of this thesis was to examine the problem of OR scheduling on the operational level. We formulated our problem from bilevel perspective, considering the head of surgeons and the individual surgeons to have different interests. Furthermore, the goal was to design and implement an algorithm that would simulate OR scheduling in a surgical department of a hospital.

We proposed and implemented a baseline integer linear programming model along with a dedicated branch-and-price algorithm. We described the master problem, the subproblem and the column generation as the essential parts of the branch-and-price algorithm. To address the bilevel optimality of this problem, we introduced two types of lazy constraints, which remove solutions that are not bilevel optimal. For the branch-and-price model, we introduced several speed-up mechanisms such as lazy constraint remembering, initial heuristics and adding multiple columns per pricing iteration.

In data analysis, we verified that our real-life dataset follows distribution parameters similar to the ones in literature (lognormal, gamma). We then created lots of instances that either came from the real-life dataset or the publicly available CHOIR dataset.

In the experiments, we mainly compared benchmarks of the ILP and branch-and-price. The ILP model seems to work decently for smaller instances with e.g. 5 or 10 surgeons and 1 operating room. Branch-and-price starts to outperform the ILP when we scale the size of instances. For example, for 4 operating rooms and 30 or 45 surgeons, the ILP timeouts every single time, whereas branch-and-price is solved to optimality for every instance. In addition, the results showed that using speed-up mechanisms can help improve the computational time. Leaving out LC remembering results in huge number of generated lazy constraints and higher time spent in the subproblem and the callbacks. The best configuration for both datasets seems to be using the advanced (assigned patients based) type of lazy constraint together with all the speed-up mechanisms: LC remembering, initial heuristics and multiple pattern generation per pricing iteration. In the last experiment, we assessed the value of game-theoretical approach. We presented two metrics to measure the quality of the equilibrium solution: the Price of Stability and the Price of Decentralization. It was shown that letting the surgeons decide their own schedule results in less efficient utilization. On the other hand, the difference in the equilibrium solution and the centralized solution was not significant.

In conclusion, this thesis is not a real production application, but rather serves as a proof of concept that shows the benefits of using branch-and-price algorithm for OR scheduling along with the lazy constraint generation as an approach to the bilevel optimization. The future work could be an improvement of the branch-and-price algorithm, possibly another speed-up mechanisms, that would make the algorithm perform better for even larger instances, e.g. bigger time scheduling horizon, more operating rooms or more surgeons.

Bibliography

Abdelrasol, Z. Y., Harraz, N., and Eltawil, A. (2013). A proposed solution framework for the operating room scheduling problems. In *Proceedings of the world congress on engineering and computer science*, volume 2, pages 23–25.

Ahmed, A. and Ali, H. (2020). Modeling patient preference in an operating room scheduling problem. *Operations Research for Health Care*, 25:100257.

Aringhieri, R., Landa, P., Soriano, P., Tànfani, E., and Testi, A. (2015). A two level metaheuristic for the operating room scheduling and assignment problem. *Computers & Operations Research*, 54:21–34.

Association, H. F. M. et al. (2003). Achieving operating room efficiency through process integration. *Healthcare financial management: journal of the Healthcare Financial Management Association*, 57(3):1–112.

Bard, J. F. and Moore, J. T. (1992). An algorithm for the discrete bilevel programming problem. *Naval Research Logistics (NRL)*, 39(3):419–435.

Brown, G., Carlyle, M., Salmerón, J., and Wood, K. (2006). Defending critical infrastructure. *Interfaces*, 36(6):530–544.

Cardoen, B., Demeulemeester, E., and Beliën, J. (2009). Sequencing surgical cases in a day-care environment: An exact branch-and-price approach. *Computers & Operations Research*, 36(9):2660–2669.

Cardoen, B., Demeulemeester, E., and Beliën, J. (2010). Operating room planning and scheduling: A literature review. *European Journal of Operational Research*, 201(3):921–932.

Chaieb, M., Sassi, D. B., Jemai, J., and Mellouli, K. (2022). Challenges and solutions for the integrated recovery room planning and scheduling problem during COVID-19 pandemic. *Medical & Biological & Engineering & Computing*, 60(5):1295–1311.

Choi, S. and Wilhelm, W. E. (2012). An analysis of sequencing surgeries with durations that follow the lognormal, gamma, or normal distribution. *IIE Transactions on Healthcare Systems Engineering*, 2(2):156–171.

Clark, P. A. and Westerberg, A. W. (1990). Bilevel programming for steady-state chemical process design—i. fundamentals and algorithms. *Computers & Chemical Engineering*, 14(1):87–97.

Doulabi, S. H. H., Rousseau, L.-M., and Pesant, G. (2016). A constraint-programming-based branch-and-price-and-cut approach for operating room planning and scheduling. *INFORMS Journal on Computing*, 28(3):432–448.

Easton, K., Nemhauser, G., and Trick, M. (2004). CP based branch-and-price. In *Constraint and Integer Programming*, pages 207–231. Springer US.

González Velarde, J. L., Camacho-Vallejo, J.-F., and Pinto Serrano, G. (2015). A scatter search algorithm for solving a bilevel optimization model for determining highway tolls. *Computación y Sistemas*, 19(1):05–16.

Hansen, P., Jaumard, B., and Savard, G. (1992). New branch-and-bound rules for linear bilevel programming. *SIAM Journal on scientific and Statistical Computing*, 13(5):1194–1217.

Harris, S. and Claudio, D. (2022). Current trends in operating room scheduling 2015 to 2020: a literature review. *Operations Research Forum*, 3(1).

Huele, C. V. and Vanhoucke, M. (2014). Analysis of the integration of the physician rostering problem and the surgery scheduling problem. *Journal of Medical Systems*, 38(6).

Jebali, A. and Diabat, A. (2015). A stochastic model for operating room planning under capacity constraints. *International Journal of Production Research*, 53(24):7252–7270.

Kamran, M. A., Karimi, B., and Dellaert, N. (2020). A column-generation-heuristic-based benders' decomposition for solving adaptive allocation scheduling of patients in operating rooms. *Computers & Industrial Engineering*, 148:106698.

Leeftink, G. and Hans, E. W. (2017). Case mix classification and a benchmark set for surgery scheduling. *Journal of Scheduling*, 21(1):17–33.

Li, F., Gupta, D., and Potthoff, S. (2015). Improving operating room schedules. *Health Care Management Science*, 19(3):261–278.

Ma, Y., Liu, K., Li, Z., and Chen, X. (2022). Robust operating room scheduling model with violation probability consideration under uncertain surgery duration. *International Journal of Environmental Research and Public Health*, 19(20):13685.

Maaroufi, F., Camus, H., and Korbaa, O. (2016). A mixed integer linear programming approach to schedule the operating room. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE.

Macario, A., Vitez, T. S., Dunn, B., and McDonald, T. (1995). Where are the costs in perioperative care?: Analysis of hospital costs and charges for inpatient surgical care. *Anesthesiology*, 83(6):1138–1144.

Maenhout, B. and Vanhoucke, M. (2009). Branching strategies in a branch-and-price approach for a multiple objective nurse scheduling problem. *Journal of Scheduling*, 13(1):77–93.

Maenhout, B., Šůcha, P., Tkadlec, O., and Nguyenova, M. T. (2023). Multi-agent short-term surgeon scheduling and surgery planning. Technical report, Czech Technical University in Prague, Ghent University, Department of Business informatics and Operations Management.

May, J. H., Strum, D. P., and Vargas, L. G. (2000). Fitting the lognormal distribution to surgical procedure times. *Decision Sciences*, 31(1):129–148.

Mazloumian, M., Baki, M. F., and Ahmadi, M. (2022). A robust multiobjective integrated master surgery schedule and surgical case assignment model at a publicly funded hospital. *Computers & Industrial Engineering*, 163:107826.

Meskens, N., Duvivier, D., and Hanset, A. (2013). Multi-objective operating room scheduling considering desiderata of the surgical team. *Decision Support Systems*, 55(2):650–659.

Milička, P., Šůcha, P., Vanhoucke, M., and Maenhout, B. (2022). The bilevel optimisation of a multi-agent project scheduling and staffing problem. *European Journal of Operational Research*, 296(1):72–86.

Momeni, M. A., Mostofi, A., Jain, V., and Soni, G. (2022). COVID19 epidemic outbreak: operating rooms scheduling, specialty teams timetabling and emergency patients' assignment using the robust optimization approach. *Annals of Operations Research*.

Paccagnan, D., Chandan, R., and Marden, J. (2022). Utility and mechanism design in multi-agent systems: An overview. *Annual Reviews in Control*.

Persson, M. J. and Persson, J. A. (2009). Analysing management policies for operating room planning using simulation. *Health Care Management Science*, 13(2):182–191.

Ponboon, S., Qureshi, A. G., and Taniguchi, E. (2016). Branch-and-price algorithm for the location-routing problem with time windows. *Transportation Research Part E: Logistics and Transportation Review*, 86:1–19.

Rahimi, I. and Gandomi, A. H. (2020). A comprehensive review and analysis of operating room and surgery scheduling. *Archives of Computational Methods in Engineering*.

Saharidis, G. K. and Ierapetritou, M. G. (2009). Resolution method for mixed integer bi-level linear problems based on decomposition technique. *Journal of Global Optimization*, 44:29–51.

Scaparra, M. P. and Church, R. L. (2008). A bilevel mixed-integer program for critical infrastructure protection planning. *Computers & Operations Research*, 35(6):1905–1923.

Sinha, A., Malo, P., Frantsev, A., and Deb, K. (2013). Multi-objective stackelberg game between a regulating authority and a mining company: A case study in environmental economics. In *2013 IEEE congress on evolutionary computation*, pages 478–485. IEEE.

Stepaniak, P. S., Heij, C., Mannaerts, G. H., de Quelerij, M., and de Vries, G. (2009). Modeling procedure and surgical times for current procedural terminology-anesthesia-surgeon combinations and evaluation in terms of case-duration prediction and operating room efficiency: a multicenter study. *Anesthesia & Analgesia*, 109(4):1232–1245.

Vanderbeck, F. (2000). On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48:111–.

Villeneuve, D., Desrosiers, J., Lübbecke, M. E., and Soumis, F. (2005). On compact formulations for integer programs solved by column generation. *Annals of Operations Research*, 139(1):375–388.

Von Stackelberg, H. (2010). *Market structure and equilibrium*. Springer Science & Business Media.

Whittaker, G., Färe, R., Grosskopf, S., Barnhart, B., Bostian, M., Mueller-Warrant, G., and Griffith, S. (2017). Spatial targeting of agri-environmental policy using bilevel evolutionary optimization. *Omega*, 66:15–27.

Wullink, G., Houdenhoven, M. V., Hans, E. W., van Oostrum, J. M., van der Lans, M., and Kazemier, G. (2007). Closing emergency operating rooms improves efficiency. *Journal of Medical Systems*, 31(6):543–546.

Zhu, S., Fan, W., Yang, S., Pei, J., and Pardalos, P. M. (2018). Operating room planning and surgical case scheduling: a review of literature. *Journal of Combinatorial Optimization*, 37(3):757–805.

	Branch-and-price					Integer linear program				
parametrization	F	F^{LPR}	$\%Gap$	$\%Opt$	Time [s]	F	F^{LPR}	$\%Gap$	$\%Opt$	Time [s]
LC=basic,$\lvert S\rvert$=10,$\lvert D\rvert$=5,$\lvert R\rvert$=1	23	22.75	0.01	1	476.24	23	22	0.04	0	1800.02
LC=basic,$\lvert S\rvert$=15,$\lvert D\rvert$=5,$\lvert R\rvert$=1	31	30.25	0.02	1	16.51	31	30	0.03	0	1800.04
LC=basic,$\lvert S\rvert$=30,$\lvert D\rvert$=5,$\lvert R\rvert$=4	81	81.00	0.00	1	43.2	81	71	0.12	0	1800.05
LC=basic,$\lvert S\rvert$=45,$\lvert D\rvert$=5,$\lvert R\rvert$=4	118	118.00	0.00	1	90.82	119	107	0.1	0	1800.05
LC=basic,$\lvert S\rvert$=10,$\lvert D\rvert$=10,$\lvert R\rvert$=1	42	34.00	0.19	0	1830.27	34	28	0.18	0	1800.03
LC=basic,$\lvert S\rvert$=15,$\lvert D\rvert$=10,$\lvert R\rvert$=1	67	52.00	0.22	0	1808.43	52	44	0.15	0	1800.02
LC=basic,$\lvert S\rvert$=30,$\lvert D\rvert$=10,$\lvert R\rvert$=4	203	201.97	0.01	0	1950.24	272	114	0.58	0	1800.11
LC=basic,$\lvert S\rvert$=45,$\lvert D\rvert$=10,$\lvert R\rvert$=4	203	189.00	0.07	0	1899.28	268	161	0.4	0	1809.02
LC=basic,average	96	91.12	0.07	0.5	1014.37	110	72.13	0.2	0	1801.17
LC=advanced,$\lvert S\rvert$=10,$\lvert D\rvert$=5,$\lvert R\rvert$=1	23	22.75	0.01	1	305.08	23	22	0.04	0	1800.02
LC=advanced,$\lvert S\rvert$=15,$\lvert D\rvert$=5,$\lvert R\rvert$=1	31	30.25	0.02	1	15.15	31	31	0	1	339.51
LC=advanced,$\lvert S\rvert$=30,$\lvert D\rvert$=5,$\lvert R\rvert$=4	81	81.00	0.00	1	53.33	81	74	0.09	0	1800.03
LC=advanced,$\lvert S\rvert$=45,$\lvert D\rvert$=5,$\lvert R\rvert$=4	118	118.00	0.00	1	43.26	118	109	0.08	0	1800.05
LC=advanced,$\lvert S\rvert$=10,$\lvert D\rvert$=10,$\lvert R\rvert$=1	35	34.00	0.03	0	1806	34	28	0.18	0	1800.03
LC=advanced,$\lvert S\rvert$=15,$\lvert D\rvert$=10,$\lvert R\rvert$=1	52	52.00	0.00	1	648.56	52	45	0.13	0	1800.04
LC=advanced,$\lvert S\rvert$=30,$\lvert D\rvert$=10,$\lvert R\rvert$=4	203	193.87	0.05	0	1805.19	228	114	0.5	0	1800.09
LC=advanced,$\lvert S\rvert$=45,$\lvert D\rvert$=10,$\lvert R\rvert$=4	203	202.10	0.00	0	1876.12	204	162	0.21	0	1800.1
LC=advanced,average	93.25	91.75	0.01	0.63	819.09	96.38	73.13	0.15	0.13	1617.48

Table 1: Benchmark comparison between different solution approaches and lazy constraints.

			Best equilibrium solution			Best decentralized solution			Best centralized solution				
α	β	scenario	Utilitation	F	Time [s]	Utilitation	F	Time [s]	Utilitation	F	Time [s]	PoD	PoS
0	1	1	0.81	2.6	1.05	0.78	3	0.92	0.83	2.6	0.91	1.15	1.00
0	1	2	0.82	3.9	0.95	0.78	4.7	0.81	0.81	3.9	0.79	1.21	1.00
0	1	3	0.82	4.6	1.79	0.78	8.1	0.94	0.82	4.6	2.32	1.76	1.00
0	1	4	0.81	2.6	1.31	0.78	3.7	1.02	0.83	2.6	0.86	1.42	1.00
0	1	5	0.82	5.1	1.98	0.77	10.3	0.87	0.82	4.7	2.95	2.19	1.09
1	0	1	0.92	13.2	1.73	0.78	34.8	0.92	0.92	13	0.93	2.68	1.02
1	0	2	0.91	14.9	3.77	0.78	35.1	0.84	0.92	13	0.96	2.70	1.15
1	0	3	0.92	18.3	1.55	0.78	37.8	0.91	0.92	18.2	1.46	2.08	1.01
1	0	4	0.91	15.1	2.51	0.79	33.7	1.05	0.92	13	0.97	2.59	1.16
1	0	5	0.91	14.7	2.05	0.77	36.7	0.88	0.92	13	0.94	2.82	1.13
1	1	1	0.92	18.3	1.56	0.78	37.8	0.91	0.92	18.2	1.46	2.08	1.01
1	1	2	0.9	23.2	6.68	0.78	39.8	0.84	0.91	22.7	2.76	1.75	1.02
1	1	3	0.91	22.4	2.3	0.78	43.1	0.87	0.91	22.4	2.04	1.92	1.00
1	1	4	0.9	19.9	4.71	0.79	37.8	1.05	0.92	18.2	1.44	2.08	1.09
1	1	5	0.89	24.7	15.27	0.77	47	0.85	0.91	23.7	4.6	1.98	1.04

Table 2: The efficiency of obtaining the equilibrium solution.