A two-phase heuristic approach to multi-day surgical case scheduling considering generalized resource constraints

Wim Vancroonenburg, Pieter Smet, and Greet Vanden Berghe

KU Leuven, Department of Computer Science, CODeS & iMinds-ITEC

3rd August 2015

Abstract

The present contribution focuses on the problem of assigning and scheduling surgical cases in rooms of an operating theatre, in order to maximize efficiency. The aim is to schedule as many surgical cases in as few operating rooms as possible, within regular operating theatre opening hours and under limited resource availability. This work generalizes many surgical case scheduling aspects considered in the literature and in practice by means of a unified resource model. The performance of a heuristic algorithm designed for this rich problem formulation is evaluated and compared on a set of real-world data. Computational results demonstrate the potential improvements obtained by using the presented approach, over schedules constructed by human planners.

Keywords: *surgical case scheduling; operating theatre scheduling; serial schedule generation; metaheuristic*

1 Introduction

Spurred by increasingly tighter budgetary constraints, hospital managers continuously aim to improve the efficiency of their most costly resources. Unsurprisingly, much attention is devoted to the operating theatre (OT), a key resource that generates significant revenue for any hospital, though at considerable costs. Macario et al. [21] indicate that surgery related services can represent more than 40% of hospital costs and revenues. Notably, Jackson [18] identifies the OT as an important profit center rather than a cost center. Being a profit center, the OT should be run to maximize throughput, while still managing added costs from over-utilization. Efficiently running an OT consisting of several operating rooms[1] (ORs) and a large surgical staff is not an easy task. It requires coordination of many different resources, both human, equipment and consumables/renewables in order to enable performing surgeries. A great deal of careful planning and organization is thus necessary to avoid delays and to ensure a high throughput. Failure to do so may

[1]Note that we use the term 'operating theatre' to refer to the general hospital unit where surgeries are performed. The term 'operating room' refers to a specific room where individual surgeries are performed.

require overtime by the surgical staff to finish all surgeries, or even cause cancellation/postponement of surgeries; ultimately resulting in revenue loss and worse quality of care.
Over the past decades, numerous software development efforts have been made to assist the planner (e.g. a surgeon, an OT manager) in scheduling the OT. These efforts have mainly computerized the process that human planners perform, e.g. providing digital user interfaces and scheduling boards in which the manual planner can allocate surgeries. Such applications can provide the planner with a good overview of availability of resources, thus improving his/her effectiveness at the task. However, the complexity of putting together a good OT schedule has not been reduced: planners still have to allocate and sequence surgeries manually (though software assisted) to find a good, workable OT schedule. Software support for automatically generating surgical schedules has seen considerably less adoption. For example, Cardoen et al. [9] report on software support for generating and optimizing surgery schedules in hospitals in Flanders, Belgium. Their survey indicates that 56% of the hospitals do not use any software support to develop and optimize their surgical schedules; 26% use software but find that the produced schedules are impractical (may contain errors, or do not consider all resources); and 11% use software that produces reliable/usable schedules (7% used other approaches, or left the question unanswered).
The present paper focuses on a decision support model and algorithm for OT scheduling, generalizing many considerations encountered in the literature and in practice. The aim is to algorithmically support OT managers in their daily/weekly task of both scheduling (determining date and time of) surgeries and assigning them to an OR. In particular, the presented approach supports scheduling other resources that may be required for performing surgeries, both human resources (e.g. the surgical staff: surgeons, anaesthesiologists, instrumenting nurses) and other resources (e.g. portable imaging tools, operating lights). To this end, generalized resource dependencies are introduced to define the dependency of surgeries on specific types of resources. A heuristic approach to this problem formulation is presented, based on a schedule generation procedure combined with local search.
The research was partly supported by Dotnext[2], a software company developing an application for managing the OT with scheduling, monitoring and reporting tools. The company's input on current OT scheduling practices and the provision of data have been essential. The remainder of this paper is organized as follows. In Section 1.1, an overview of previous research on the operating theatre scheduling problem is provided. Next, in Section 2 the proposed model is described, with Section 3 detailing the heuristic approach to operating theatre scheduling. Two example models addressing important considerations from practice are presented in Section 4, in order to emphasize the flexibility of the resource model. Computational experiments in Section 5 show that the approach scales favourably with problem size, and is able to schedule surgical cases in a large hospital setting in limited time. Finally, Section 6 concludes the paper.

1.1 Related work

Given the central role of the OT in any hospital, and the impact it has on hospital costs and revenue, it has been the subject of a myriad of studies. In particular,

[2]Dotnext, Dikkemeerweg 172, 1652 Alsemberg, Belgium – `http://www.dotnext.be`

optimization and decision support for planning and scheduling in OTs is not at all new. Cardoen et al. [8] review 115 studies on the matter published after 2000, and many more have been published since 2010. The studies can be categorized based on the decision level they focus on [17]. The strategic and tactical decision levels are concerned with long resp. mid term decision making, mostly determining service and capacity levels (strategic), and allocating capacity (tactical). The *operational decision level* focuses on short term decisions involving execution of the service process. The surgical scheduling problem considered in this paper is situated at this level and is concerned with scheduling individual surgical cases in an OT over a certain planning horizon. The literature overview therefore concentrates on operational decision making. However, one particular decision at the tactical level is of interest for this work: the OT planning strategy. This strategy determines how OT capacity is distributed among different surgeons or surgeon groups. Roughly, three different strategies can be identified [12, 14]: an open scheduling strategy, a block scheduling strategy and a modified block scheduling strategy.

- *Open scheduling strategy*: OT capacity is not reserved for surgeons or surgeon groups. Rather, the OT capacity is managed as a single, shared entity to which surgical cases can be assigned.

- *Block scheduling strategy*: preallocates the OT capacity to the different surgeons or surgeon groups. OT capacity is divided into blocks or slots consisting of an OR for a specified duration (usually a half day or full day). A block schedule, also denoted as the master surgical schedule (MSS), determines which blocks are allocated to which disciplines for each day of the week. Surgeons are free to assign surgical cases to their allotted blocks as they see fit.

- *Modified block scheduling*: complements block scheduling with a policy to check for underutilization. If underutilization of an upcoming OR block is likely, this block may be opened to other surgeons for assigning surgical cases.

A block schedule or MSS is relatively common in practice, as it has the advantage of establishing a (semi-)fixed situation, resulting in a stable flow and consistent mix of patients. It also reduces the scheduling complexity since surgeries can only be allocated to blocks of their particular medical discipline. An open scheduling strategy, on the other hand, may result in higher performance and throughput due to increased scheduling flexibility.
The construction of the MSS has been shown to have significant impact on other downstream resources (bed usage, workload) as it partly determines the arrival rate and arrival pattern of surgical patients. Therefore, these resources should not be ignored when constructing/updating the MSS. Beliën and Demeulemeester [4] showed how to construct an MSS that results in an expected levelled bed occupancy, by minimizing a weighted sum of the maximum expected bed occupancy and the maximum expected variance of the bed occupancy. van Essen et al. [32] also showed how to relate downstream ward bed usage to the MSS. By doing this, they were able to reduce the number of required beds in HagaZiekenhuis (Den Haag, the Netherlands) by rearranging the MSS. Next to bed occupancy, the MSS also influences the workload in nursing wards. Therefore, it is worthwhile to consider staffing decisions in the process of constructing the MSS. Beliën and Demeulemeester [5] showed how to match and integrate the construction of an MSS with nurse scheduling.

Considering OT scheduling and planning decisions at the operational level, the present paper emphasizes scheduling individual surgical cases. Many studies further decompose this process into two steps, denoted *advance scheduling* and *allocation scheduling* [22, 27]. Other authors, e.g. [29], have coined terms such as "intervention assignment" and "intervention scheduling", and "surgical case assignment" [1] and "surgical case scheduling" [7]. In any case, the former process deals with assigning a surgery date, and possibly an OR, to individual surgical cases in an upcoming planning period. The latter process deals with sequencing/scheduling individual surgical cases throughout the day within the different ORs, determining a specific start time for each surgery. Note that there is no clear distinction between these two steps. Depending on the hospital's policy, some flexibility may be allowed in the allocation scheduling step: i.e. OR assignments may be changed, surgeries may be cancelled or postponed.

Concerning the advance scheduling problem, most studies focus solely on the OT, assigning surgical cases to a specific surgery day and operating room. This is often a weekly process of selecting surgical cases from a waiting list, and possibly (pre-) assigning them to individual ORs. The main objectives typically involve minimizing overtime and underutilization of ORs [12, 19], as well as patient related costs and quality of service measures, such as waiting time [15, 19] and tardiness with respect to due dates [29].

The allocation scheduling problem follows the advance scheduling problem, and takes as input the surgical cases for the upcoming planning period. The main goal of the scheduling process is to construct a feasible work plan for each surgery day. Therefore, the allocation scheduling process typically considers more resources and more operational constraints in order to be feasible in practice. Examples are surgeons operating in multiple rooms [23, 25]; the capacity-limited post anaesthetic care unit (PACU) [7, 25] or the intensive care unit (ICU); (sequence-dependent) setup times between surgeries [28, 35]. Many studies target optimization of a variety of performance measures. Minimizing overtime or makespan are of main importance.

Some studies handle both the advance scheduling problem and the allocation scheduling problem in one single approach. For example, Marques et al. [23] presented an integer programming model to both select surgical cases to be performed in the upcoming week and determine an OR and start time (slot). Riise and Burke [29] addressed a combined surgery allocation and surgery admission planning problem, considering patient waiting time and tardiness, and surgeon overtime. More recent studies have broadened the scope of the problem even further, considering for example the impact of the surgical schedule on downstream resources (e.g. bed usage at nursing wards [3]), or integration with personnel scheduling related issues [33].

Yet another discriminating feature of approaches is the consideration of uncertainty in surgery durations and emergency interventions. Clearly, uncertainty is an important issue due to the highly variable nature of surgical cases. Denton et al. [11] presented a stochastic model for surgery sequencing and scheduling in a single OR. Their aim is to minimize a weighted objective function consisting of waiting time, idle time and tardiness. Hans et al. [16] presented a robust surgery loading study for the advance scheduling problem. Overtime should be minimized and free capacity should be maximized, whilst considering uncertainty in surgery durations and varying flexibility with respect to a base schedule. They minimize the required slack for avoiding overtime by exploiting the *portfolio effect* – a decrease in risk by increasing diversity (by means of non-correlated portfolio components). Min and

Yih [26] introduced a stochastic approach to the advance scheduling problem that considers both uncertainty on surgery durations, emergency arrivals and length of stay in the ICU. Their objective was to minimize block overtime and patient waiting time related costs. Bruni et al. [6] presented a heuristic approach to a stochastic programming model for the advance scheduling problem. They consider different (recourse) methods to deal with infeasibilities in the surgical schedule caused by the realization of uncertain parameters. On a weekly basis, an advance schedule is constructed, maximizing an abstract priority weighted profit of performing surgeries, decreased by the expected recourse cost.
Lastly, studies differ in the approaches used for solving the respective models. Many studies develop a mathematical programming formulation (e.g. Mixed-Integer Linear programming, Constraint programming) of the scheduling problem [2, 3, 12, 13, 20, 23, 25, 30, 34], which can be implemented and solved with dedicated software. However, often a heuristic method (possibly based on the mathematical programming approach) is developed alongside [12, 13, 34] to limit the execution time of the approach when dealing with realistic-size instances. Several metaheuristic methods have been developed as well, such as genetic algorithms [24, 30] and local search based algorithms [2, 3, 16, 29].

1.2 Contribution

The main motivation for this work was to develop a sufficiently general and flexible approach to deal with different policies to OT scheduling across hospitals. In this study, a decision support model and approach for the surgical scheduling problem is developed that generalizes many considerations pointed out in literature and in practice: employing multiple ORs, assigning a surgical team, material dependencies for surgical cases, etc. A generalized resource dependency model is proposed for specifying dependencies of surgical cases on different resources.
The model of Meskens et al. [25] is one of the few considering many aspects of surgical case scheduling encountered in practice, targeting, similar to the present paper, a general decision support tool. Meskens et al. present a modular model for the daily surgical case scheduling problem using constraint programming, in which different considerations are grouped into modules and can be turned 'on/off' according to the application's requirements. It does, however, not accommodate several considerations from practice, such as minimizing resource transfers (avoiding that nurses, anaesthesiologists have to move between rooms); the concept of *surgical phases* (a surgeon should only be present during an incision phase); multi-day scheduling.
Many studies [e.g. 7, 23–25] employ a discrete representation of time. Both the planning horizon and the surgical case durations are discretized into multiples of a certain time unit. To keep model sizes tractable, time unit sizes are often in the order of 15-30 minutes. As surgical durations must be rounded due to this discretization, underutilization may be introduced (assuming rounding up to the nearest multiple of a time slot). Contrastingly, the present study relies on a continuous time representation, enabling any scheduling precision and thus avoiding underutilization due to limited precision. Moreover, the algorithm's complexity does not depend on the precision of time.
Finally, an important practical contribution is that the approach has been designed for implementation in an OT management application of a software company. The

generality and flexibility of the model and approach are essential in this aspect, as the software application must be able to cope with the diversity of all its end users.

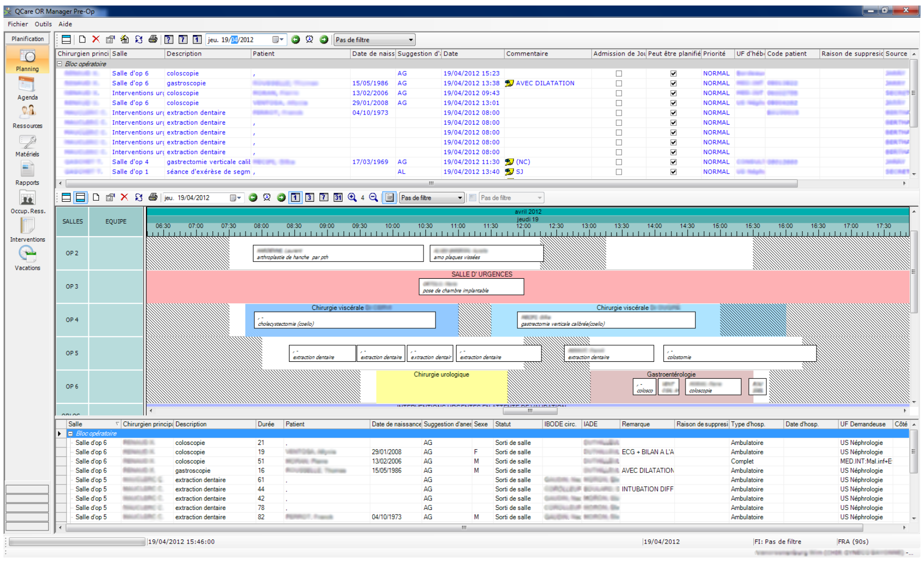
2 Surgical case scheduling problem

2.1 Problem statement

The problem addressed in this paper is largely motivated by the goal to have a working decision support approach for surgical case scheduling, suited for integration in hospital applications. The research project with the software company Dotnext was set up to reach this goal and has strongly influenced modelling and algorithm development. The software application QCare OR, developed by Dotnext (before the start of the project), allows to organize and support the complete operative process, from pre-operative planning, to online monitoring of current status, to finally post-operative planning and reporting. QCare OR is intended for use in different hospitals and customers in different countries; it is therefore set up quite flexibly to accommodate different organisational approaches to manage the operating theatre. Figure 1 shows the QCare OR interface for pre-operative planning, which is the focus of the present paper. The application allows to configure operating rooms and resources (personnel, equipment) along with their availability (which may vary per OR /resource). Surgical appointments can be created with their expected duration and various resource dependencies, e.g. a surgeon, instrumenting nurse, anaesthesiologist. These appointments can then be assigned to specific days, and scheduled in specific operating rooms at specific times. Configuration of the application is typically done at installation, and when ORs/resources are introduced, modified or removed. The scheduling process is performed on a daily or weekly basis (depending on the hospital policy). Often times, the schedule is updated during the day/week to reflect changes, e.g. new arrivals, emergencies, unplanned personnel unavailability.
To algorithmically support this manual scheduling process, we formulate it as a multi-day surgical case scheduling problem with resource dependencies. In what follows, we define the most relevant elements of the problem.

- **Operating room:** An operating room is dedicated to performing surgeries on individual patients. It is typically one room out of a larger set of operating rooms in the operating theatre, dedicated to performing surgeries. An operating room provides several (static) tools and equipment for supporting the surgical process. Quite often, operating rooms in an operating theatre are equipped differently, making some of them more suitable for certain types of surgeries.

- **Resources:** Next to the static equipment provided in operating rooms, certain equipment or tools are not room-bound and may be used in different rooms. For example, mobile surgical lamps or imaging equipment may be moved between operating rooms and support different surgeries across rooms. Apart from tools and equipment, personnel and the surgical staff can also be considered as important resources of the operating theatre. The surgical staff are among the most crucial and capacity-limited resources for performing surgeries and must be accounted for when scheduling surgeries.



Chirurgien princi	Salle	Description	Patient	Date de naiss	Suggestion d'	Date	Commentaire	Admission de Jou	Peut être planifié	Priorité	UF d'héb	Code patient	Raison de suppresio	Source
Bloc opératoire														
[redacted]	Salle d'op 6	coloscopie	,		AG	19/04/2012 15:23		☐	✓	NORMAL	[redacted]			[redacted]
[redacted]	Salle d'op 6	gastroscopie	[redacted]	15/05/1986	AG	19/04/2012 13:38	AVEC DILATATION	☐	✓	NORMAL	[redacted]	[redacted]		[redacted]
[redacted]	Interventions urg	coloscopie	[redacted]	13/02/2006	AG	19/04/2012 09:43		☐	✓	NORMAL	[redacted]	[redacted]		[redacted]
[redacted]	Salle d'op 6	coloscopie	[redacted]	29/01/2008	AG	19/04/2012 13:01		☐	✓	NORMAL	[redacted]	[redacted]		[redacted]
[redacted]	Interventions urg	extraction dentaire	[redacted]	04/10/1973		19/04/2012 08:00		☐	✓	NORMAL		[redacted]		[redacted]
[redacted]	Interventions urg	extraction dentaire	,			19/04/2012 08:00		☐	✓	NORMAL				[redacted]
[redacted]	Interventions urg	extraction dentaire	,			19/04/2012 08:00		☐	✓	NORMAL				[redacted]
[redacted]	Interventions urg	extraction dentaire	,			19/04/2012 08:00		☐	✓	NORMAL				[redacted]
[redacted]	Interventions urg	extraction dentaire	,			19/04/2012 08:00		☐	✓	NORMAL				[redacted]
[redacted]	Salle d'op 4	gastrectomie verticale calit	[redacted]	17/03/1969	AG	19/04/2012 11:30	(NC)	☐	✓	NORMAL	[redacted]	[redacted]		[redacted]
[redacted]	Salle d'op 1	séance d'exérèse de segm	,		AL	19/04/2012 13:40	SJ	☐	✓	NORMAL	[redacted]			[redacted]

Salle	Chirurgien princip	Description	Durée	Patient	Date de naissance	Suggestion d'anes	Sexe	Statut	IBODE circ.	IADE	Remarque	Raison de suppresi	Type d'hosp.	Date d'hosp.	UF Demandeuse	Côté
Bloc opératoire																
Salle d'op 6	[redacted]	coloscopie	21	.		AG		Sorti de salle		[redacted]			Ambulatoire		US Néphrologie	
Salle d'op 6	[redacted]	coloscopie	19	[redacted]	29/01/2008	AG	F	Sorti de salle		[redacted]	ECG + BILAN A L'A		Ambulatoire		US Néphrologie	
Salle d'op 6	[redacted]	coloscopie	51	[redacted]	13/02/2006	AG	M	Sorti de salle		[redacted]			Complet		MED.INT:Mal.inf+E	
Salle d'op 6	[redacted]	gastroscopie	16	[redacted]	15/05/1986	AG	M	Sorti de salle		[redacted]	AVEC DILATATION		Ambulatoire		US Néphrologie	
Salle d'op 5	[redacted]	extraction dentaire	61	.		AG		Sorti de salle	[redacted]	[redacted]			Ambulatoire		US Néphrologie	
Salle d'op 5	[redacted]	extraction dentaire	44	.		AG		Sorti de salle	[redacted]	[redacted]	INTUBATION DIFF		Ambulatoire		US Néphrologie	
Salle d'op 5	[redacted]	extraction dentaire	42	.		AG		Sorti de salle	[redacted]	[redacted]			Ambulatoire		US Néphrologie	
Salle d'op 5	[redacted]	extraction dentaire	78	.		AG		Sorti de salle	[redacted]	[redacted]			Ambulatoire		US Néphrologie	
Salle d'op 5	[redacted]	extraction dentaire	82	[redacted]	04/10/1973	AG	M	Sorti de salle	[redacted]	[redacted]			Ambulatoire		US Néphrologie	

19/04/2012 15:46:00 | 19/04/2012 | FI: Pas de filtre | FRA (90s)

Figure 1: Dotnext's QCare OR pre-operative planning user interface. The interface for pre-op planning is divided into three sections: waiting list of appointments to be scheduled in the current planning period (top); scheduling dashboard showing the assigned OR, starting time and duration of scheduled surgeries (middle); detailed information and resources assigned to the scheduled surgeries (bottom).

- **Surgical cases:** Surgical cases are the main element of the problem under consideration. After a consultation, determining that a patient must undergo surgery, a surgery appointment is made. The surgeon may then record the type of surgery, how long it is expected to last and what kind resources and surgical staff are required. The surgery date and start time depend on the availability of a suitable operating room, as well as the required resources and surgical staff. Resources or staff often need not be present for the entire duration of the surgery. A supervising surgeon, for example, may only be present during a crucial part of the surgery, whereas the remainder is handled by the other surgical staff (e.g. another assisting surgeon).

The manual planner has a list of surgical appointments to be scheduled in the upcoming period. The goal is to schedule all surgical cases within this period. However, due to limited capacity of ORs and availability of resources, this may not be possible without incurring overtime. Dealing with overtime may vary greatly among hospitals. Some hospitals have a policy to never schedule elective cases in overtime, while others may allow it, provided it is cost-effective. We opt in this study to schedule surgical cases only within normal capacity. This may leave surgical cases unscheduled. The main motivation for this approach is that hospitals can always decide manually if it is necessary to perform the remaining surgeries during overtime.

2.2 Model formulation

The surgical case scheduling problem deals with scheduling a set of surgical cases S in a set of ORs O, over a finite planning horizon (represented by a set D of one

Set	Index	Description
S	$s = 1, \dots \lvert S \rvert$	Set of surgical cases
O	$o = 1, \dots \lvert O \rvert$	Set of ORs
D	$d = 1, \dots, \lvert D \rvert$	Set of days in planning horizon
RT	$j = 1, \dots, \lvert RT \rvert$	Set of resource types
R	$r = 1, \dots, \lvert R \rvert$	Set of resources
$RT_r \subseteq RT$		Subset of resource types that resource r can be assigned to.
$R_j \subseteq R$		Subset of resources of type j. Note that R_{j_1} and R_{j_2} are not necessarily disjoint if $j_1 \neq j_2$: some resources are flexible and can fulfil different resource type dependencies (e.g. nursing staff who have multiple skills).
$R^{TRAN} \subseteq R$		Set of resources for which transfers between ORs must be minimized.
$R^{IDLE} \subseteq R$		Set of resources for which idle time must be minimized.
$AV_{o,d}$		Set of availability intervals $[\text{start}, \text{end})$ for OR o on day d.[1]
$AV_{r,d}$		Set of availability intervals $[\text{start}, \text{end})$ for resource r on day d.
$O_s^1, O_s^2, O_s^3 \subseteq O_s$		Set of preferred, possible and *if-necessary* operating rooms for surgery s
$RT_s^r, RT_s^o \subseteq R$		Set of required (optional) resource types for surgery s
$D_s \subseteq D$		Subset of planning horizon on which case s can be scheduled

[1] $[a, b)$ denotes the half-open interval with limits a (inclusive) and b (exclusive).

Table 1: Summary of input sets and indices.

or more days). In this work, an *open-scheduling* policy is assumed; thus allowing surgical cases to be scheduled in any OR at any time. Fei et al. [12] also noted that an open scheduling strategy is more general than block scheduling. Schedules complying with a block scheduling strategy, also fit an open scheduling strategy. Nevertheless, the approach can also be adapted to accommodate block scheduling quite easily (see Section 3.1).

The aim is to schedule as many surgical cases in as few ORs as possible, within the regular opening hours of the OT. Furthermore, a set of hard resource and ordering constraints must be considered. Soft resource constraints and desiderata are penalized if violated or not met. In what follows, the elements and constraints of the problem are described. Notation will be introduced along the description but is also summarized in Tables 1 and 2.

Parameter	Unit/Possible values	Description
C_r	dimensionless, $C_r \in \mathbb{N}_0$	Number of ORs resource r can be used in, per day.
$AF(r_1, r_2)$	positive, negative, neutral	Affinity between resources r_1 and r_2.
p_s	dimensionless, $C_r \in \mathbb{N}_0$	Priority of case s
d_s	unit of time (e.g. minutes)	Surgical duration of case s
s_{sj}	unit of time (e.g. minutes)	Start of surgical phase (offset from start of case s in the OR) of resource type j
d_{sj}	unit of time (e.g. minutes)	Duration of surgical phase of resource type j
C_{sj}	dimensionless, $C_r \in \mathbb{N}_0$	Number of resources of resource type j required for surgical case s

Table 2: Summary of model parameters.

2.2.1 Basic problem

The aim of this work is to schedule as many surgical cases in an OT, over a specified period, as possible. In practice it is common to schedule the surgical cases for the upcoming week, one week in advance. However, it is not uncommon that certain arrangements have already been made with respect to the surgery date of individual cases. Therefore, this work considers a general setting where surgical cases $s \in S$ are eligible to be scheduled on one day from a set of possible days $D_s \subseteq D$. Clearly, a surgical case should only be scheduled once:

Constraint 1. *A surgical case s can be scheduled on at most one day $d \in D_s$.*

It is assumed that an open scheduling policy is maintained, i.e. surgical cases can be scheduled in any OT. However, practice may differ: some surgical cases may only be performed in certain ORs, due to restrictions on capacity, fixed equipment, etc. Therefore, for each surgical case s a set of suitable ORs O_s is considered. Obviously:

Constraint 2. *A surgical case s can be scheduled in at most one OR $o \in O_s$.*

In addition, some ORs may be more suited than others for a surgical case s. Therefore, a distinction is made between ORs. “Preferred” rooms ($O_s^1 \subseteq O_s$) are best suited for performing surgery for case s, but “possible” rooms ($O_s^2 \subseteq O_s$) and “if-necessary” rooms ($O_s^3 \subseteq O_s$) may be used as well. “If-necessary” rooms should be avoided unless no other room is available. Thus:

Soft constraint 1. *Surgical cases should be scheduled in “preferred” rooms as much as possible, but “possible” and “if-necessary” rooms may be used as well. “If-necessary” rooms should be avoided.*

Each surgical case s has a specific duration d_s during which the case occupies the OR. For the purpose of presentation, we assume that the precision of the duration d_s is specified in minutes, as well as any other measure of time (i.e. availability intervals, see further). However, as it will be shown in Section 3, any precision of time is possible without loss of performance.
Clearly the following must hold:

Constraint 3. *A surgical case s cannot be overlapping in time with any other surgical case s' scheduled in the same OR.*

ORs are only ‘open’ during specific time windows, often only from 7-8 am until 5-6 pm. Sometimes an OR may also be closed during lunch. Therefore, availability intervals can be specified for each OR and each day d as $[\text{open}, \text{close}) \in AV_{o,d}$, and:

Constraint 4. *A surgical case s can only be scheduled on day d in OR o within an availability interval $[open, close) \in AV_{o,d}$.*

Note: *To accommodate block scheduling, availability intervals can be expanded to also indicate to which surgical discipline they belong to. Surgical cases should also indicate the surgical discipline they belong to.*

2.2.2 Resource dependencies

Surgical procedures may require the presence of some resources. In this problem setting, resource dependencies are considered in a broad sense: both human resources such as surgeons, anaesthesiologists, nurses, but also specific material such as portable imaging equipment, surgical lights or other tools.

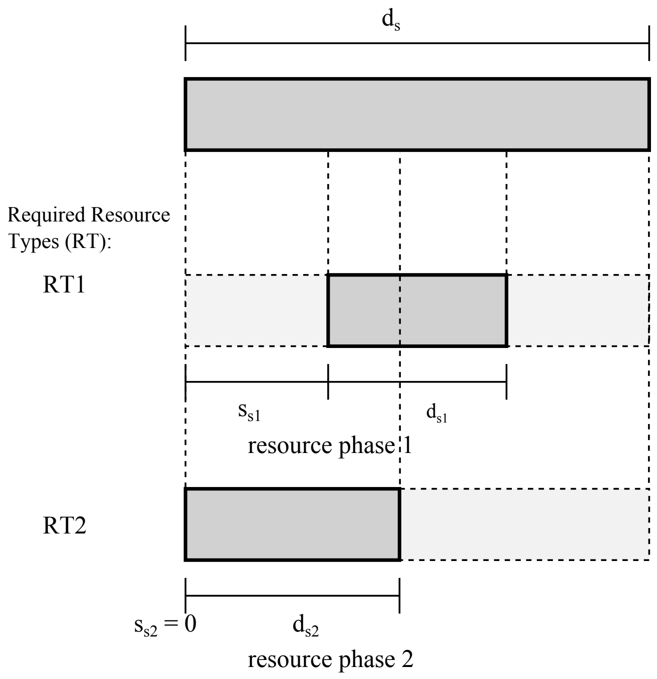


Figure 2: Definition of a resource phase. For surgical case s, one dependency on a resource type (RT1) is defined, with offset from start s_{s1} and duration d_{s1}.

We distinguish between resource types and resources. Resource types (denoted $j \in RT$) represent general types such as surgeons, nurses, instrumenting nurses, anaesthesiologists, lamps, imaging devices. That is, they represent a specific functionality (for materials) or role (for people). Resources (denoted $r \in R$) on the other hand represent true physical resources that have specific functionalities or can perform certain roles. The set of resource types a resource r belongs to is denoted by $RT_r \subseteq RT$, and the set of resources of a certain resource type j is denoted $R_j \subseteq R$.
Each surgical case specifies dependencies on resource types rather than specific resources, and are denoted by RT_s. For each resource type dependency $j \in RT_s$, a count C_{sj} specifies the number of resources required for a specific resource type. Therefore, an additional complexity in this model is that for each resource type dependency $j \in RT_s$ of a surgical case s, sufficient resources $r \in R_j$ must be assigned. Note that in some hospital settings, the surgeon/surgical staff may already be assigned to the surgical case beforehand. This is also easily accommodated by making resource types for each individual resource. By doing so, a resource dependency on a specific resource can be defined.
Furthermore, resources may only be essential during a specific part of the surgical case duration. For example, an imaging tool may only be needed at the start of a surgical case. A (supervising) surgeon on the other hand may only be present during the incision phase of the surgical case. Therefore, each resource type dependency $j \in RT_s$ a surgical case s depends on, also specifies a *surgical phase* by an offset s_{sj} from the scheduled start, and a duration d_{sj} (Figure 2). Resources only need to be assigned during this surgical phase, rather than during the entire surgical case. Additionally, this work limits resource assignments to one surgical phase

per surgical case.
As with ORs, resources also have limited availability. Thus, for each resource r availability intervals are specified for each day d of the scheduling period as $[\text{start}, \text{end}) \in AV_{r,d}$.
Finally, a distinction is made between required and optional resources (denoted RT_s^r and RT_s^o respectively); the former being necessary for scheduling the surgical case, while the latter are preferably present. Such a distinction may be made, for example, if it is important for the human planner to schedule surgical cases even when resource availability is insufficient. Another possible use case is to define resource dependencies that are not medically relevant but preferably present. For example, in a teaching hospital it may be interesting to assign a student surgeon to overview the surgery, but it may not be required to perform the surgery.
The resource constraints are thus:

Constraint 5. *For each required resource type $j \in RT_s^r$ a surgical case s may depend on, C_{sj} resources $r \in R_j$ should be assigned during $[s_{sj}, s_{sj} + d_{sj})$.*

Constraint 6. *A resource can only be assigned to one surgical case at any given time and is used during the entire surgical phase it is assigned to (no pre-emption).*

Soft constraint 2. *Optional resource types RT_s^o for a surgical case s must be assigned as much as possible, i.e. shortages with respect to C_{sj} should be minimized.*

2.2.3 OR and resource considerations

Resource efficiency
Next to handling the dependency of surgical cases on resources, the aim is also to schedule resources assigned to surgical cases as efficiently as possible. Therefore some measures of resource efficiency are considered as well.
Firstly, resources considered in this paper are assumed to be *mobile*. This is a reasonable assumption given that stationary resources are fixed to a specific OR, and are thus scheduled implicitly together with the OR. Even though resources are assumed mobile, some should only be used in few OTs, and some should not be transferred too much (e.g. large equipment). Therefore, for such resources the aim is to minimize the number of ORs they are used in, and the transfers. However, other resources may not have such a restriction. A surgeon assigned to two ORs, may want to alternate surgical cases in different rooms to avoid non-surgical tasks that occur at the start/end of a surgery, such as anaesthesia and closing/cleaning. Thus, transfers for such resources (i.e. surgeons, but also others) should not be constrained/minimized, but the number of ORs these resources are used in should not exceed their assigned ORs.
To accommodate this, the following is defined:

- C_r : the number of ORs a resource r may be used in.
- $R^{TRAN} \subseteq R$: the set of resources for which transfers between ORs must be minimized.

These dependencies are then formalized by:

Soft constraint 3. *The number of ORs a resource r is used in, should be smaller than C_r, i.e. the surplus should be minimized.*

Soft constraint 4. *For resource $r \in R^{TRAN}$, transfers between ORs should be minimized.*

Secondly, next to transfers between ORs it may also be important that resources are not left idle between surgical cases. For example, surgeons may prefer that their surgical cases are not scattered during the day, but are grouped together. Idle time between surgical cases may need to be minimized, for ensuring maximal efficient usage of a resource. Let $R^{IDLE} \subseteq R$ denote the set of resources for which the idle time between surgical cases must be minimized. Then:

Soft constraint 5. *The total idle time between surgical phases to which r is assigned should be minimized, for all $r \in R^{IDLE}$.*

Resource affinity cost
Meskens et al. [25] point out possible affinities between surgical staff members, i.e. some people work better together than others. Their model considers a positive-valued affinity matrix defined between individual surgical staff members to address this aspect. Affinities are ranked on a range of 0 to 9, with 0 denoting incompatible and 9 denoting strong preference. A simplified idea applied in this research distinguishes between negative, neutral and positive affinities. Therefore, an affinity cost matrix $AF(r_1, r_2)$ with $r_1, r_2 \in R$, is defined as follows (assuming minimization of conflicts):

$$AF(r_1, r_2) = \begin{cases} 1 & \text{if } r_1, r_2 \text{ have a } \textit{negative} \text{ working affinitiy,} \\ -1 & \text{if } r_1, r_2 \text{ have a } \textit{positive} \text{ working affinity,} \\ 0 & \text{if } r_1, r_2 \text{ have a neutral working affinity.} \end{cases} \tag{1}$$

The aim is consider these affinity costs when assigning resources to surgical cases.

Soft constraint 6. *Positive working affinities should be maximized when assigning resources to surgical cases, whereas negative working affinities should be avoided. I.e. the total affinity cost should be minimized.*

Note that no strong incompatibility is defined. The approach does not model completely incompatible resources. It is only considered as a soft constraint (violations are thus possible, but should be minimized).

OR idle time
Maximal throughput and efficient occupation of ORs are also ensured, by minimizing idle time between surgeries scheduled in an OR.

Soft constraint 7. *The total idle time between surgeries should be minimized for every OR.*

Note that this goal does not leave time between surgeries for e.g. cleaning the OR. We assume that such 'setup times' are considered in the surgical case duration d_s. Surgical phases can be defined, accordingly, to end when cleaning should start.

Ordering constraints
For medical and practical reasons, a preferential ordering of surgical cases exists within any OR. For example, patients with latex allergies are treated before other surgical cases, while patients who may be infectious are operated on after all other

cases. Another common case is that children are operated earlier during the day, after which adults follow. This work assumes that such sequencing rules can be captured by an ordering priority p_s, specific to each case s, and:

Constraint 7. *Within any OR, surgical cases should be performed in order of increasing priority p_s.*

2.2.4 Objective function

The main aim of the surgical case scheduling problem presented in this work, is scheduling as many surgical cases as possible (within availabilities). However, this may have a side effect in case of low availability (and surgeries are left unscheduled). In such a case, the algorithm will be biased towards scheduling surgeries with short durations since more of these can be scheduled. To counter this bias, rather than minimizing the number of unscheduled surgeries, the total duration of the unscheduled cases is minimized. As a secondary objective, the number of OR days, i.e. days individual ORs are occupied, is minimized. Essentially the combination of these first two objectives maximize efficient usage of the OT.
In addition, soft constraints 1-7 should also be considered. Therefore, we propose a lexicographic optimization objective function with the following ordering of importance:

1. Minimize the total surgical duration of surgical cases left unscheduled.
2. Minimize the number of OR days of the schedule.
3. Minimize the number of surgical cases assigned to *InIfNecessary* rooms (**Soft constraint 1**).
4. Minimize the number of optional resources left unassigned (**Soft constraint 2**).
5. Minimize the number of resource 'overloads': resources r assigned to surgical cases in more than C_r rooms (**Soft constraint 3**).
6. Minimize the number of resource transfers of resources $r \in R^{TRAN}$ (**Soft constraint 4**).
7. Minimize the total affinity cost (**Soft constraint 6**).
8. Maximize surgical case assignments to *Preferred* rooms (**Soft constraint 1**).
9. Minimize operating room idle time between surgical cases (**Soft constraint 7**).
10. Minimize resource idle time between surgical case assignments (**Soft constraint 5**).

Note that this ordering of importance is essentially arbitrary. The ordering depends on a hospital's preference. However, as a decision support tool intended to *construct* schedules it is important to minimize the total duration of unscheduled surgeries as primary objective. Given the objectives presented, it would be easy to find a schedule without soft constraint violations by leaving surgical cases unscheduled.

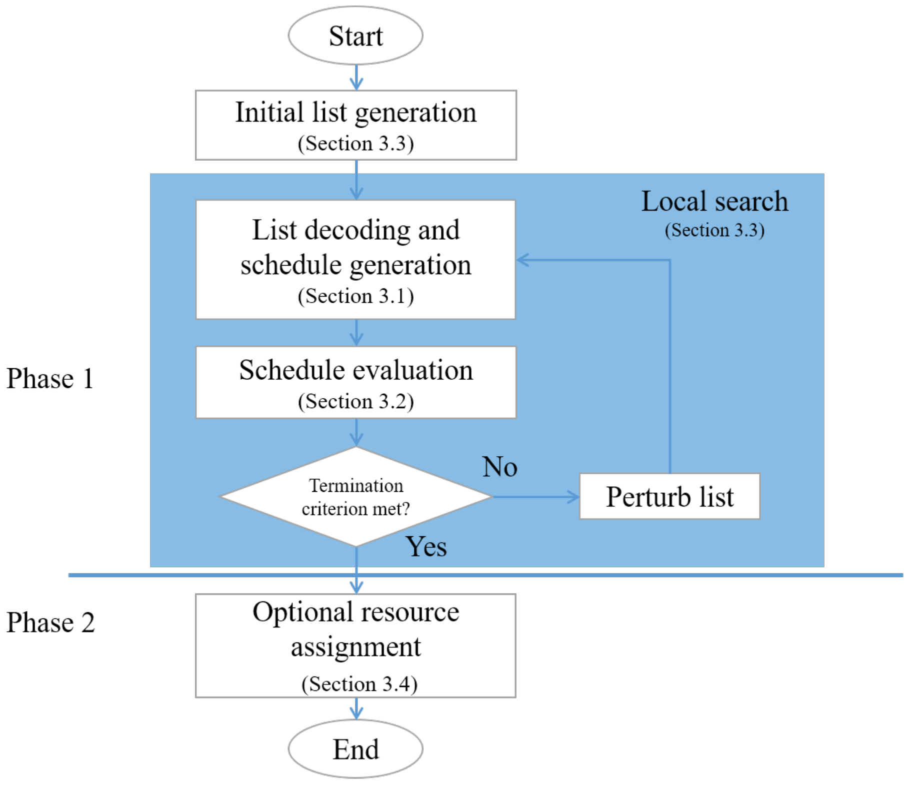


Figure 3: General overview of the heuristic approach.

3 Algorithmic approach

As already mentioned, earlier research efforts have presented approaches based on mathematical programming formulations and related techniques (e.g. integer programming [23], column generation [7]) and constraint programming [25, 35]. Such approaches often suffer from the dimensionality of the problem, limiting scalability. Heuristic methods have been developed alongside to tackle realistic-size instances. In this study we have opted for a heuristic two phase approach in order to deal with the generalized problem definition, which may involve many resources.
An overview of the two phase approach is presented in Figure 3. The approach is based on a list decoding procedure for generating feasible schedules. This procedure takes as input a list of surgical cases in a specific order and produces a schedule that adheres to all hard constraints of the problem. Next, a local search algorithm is used to manipulate this list of surgical cases in order to find new feasible schedules of better quality. Finally, in a second phase, optional resources are greedily assigned to surgical cases.
The main motivation for this two phase approach is exactly due to the dimensionality of the problem, in the context of our case study presented in Section 5. In this setting, optional resources are quite numerous and considering them in the first phase will slow down the local search algorithm too much; ultimately limiting its interactive usage.

Notation	Description
(s, o, d, AR)	Tuple assigning s to room o on day d, with resource assignments AR.
$AR = \{(j, r) \mid j \in RT^r_s, r \in R_j\}$	Set of resources r assigned to resource type j for case s.
$T_{o,d}, T_{r,d}$	Interval tree of surgery intervals/surgical phase intervals for OR o /resource r on day d.
$T(s)$	Starting time of case s.
$T_r(s)$	Starting time of resource r for surgical case s.

Table 3: Summary of data structures and decision variables.

3.1 Indirect solution representation and list decoding procedure

The list decoding procedure takes as input an ordered list of surgical cases in which each surgical case is already assigned to an OR o, a day d and to a set of resources AR. Based on this list, a feasible schedule/solution is constructed. The main decision variable in this approach is an ordered list of tuples, each consisting of a surgical case s, an OR o, an assigned day d and a set of assigned required resource type/resource pairs $AR = \{(j, r) \mid j \in RT^r_s, r \in R_j\}$, or formally:

$$< (s_1, o_1, d_1, AR_1), (s_2, o_2, d_2, AR_2), \ldots, (s_{|S|}, o_{|S|}, d_{|S|}, AR_{|S|}) > \tag{2}$$

Note that only required resource types are considered for resource assignments. Optional resources are handled differently, see Section 3.4.

The ordered list must be feasible with respect to the ordering/priority constraint (Constraint 7) for any day d and OR o, i.e. (s_k, o_k, d_k, AR_k) should appear earlier in the list than (s_l, o_l, d_l, AR_l) if $d_k = d_l, o_l = o_k$ and $p_{s_k} < p_{s_l}$ for any $1 \leq k < l \leq |S|$. Given this priority feasible list, the list decoding procedure produces a feasible schedule as follows.

1. **Initialize data structures**: the most important data structures are representations of a schedule for both the ORs and the resources. These data structures will hold the partial schedules of all surgical cases/surgical phases assigned to an OR/resource.

   A schedule is implemented as an *interval tree*, a balanced binary tree data structure for storing intervals over the real numbers. Intervals are stored in the nodes of the tree, ordered first by increasing start time and second by increasing end time (when start times are equal). Additionally, each node maintains both the min and max interval value stored in the subtree rooted at the node. Figure 4 shows an example interval tree.

   An interval tree is well suited for testing whether a point/interval is contained in/overlaps with an interval in the tree; such operations can be performed in $O(\log n)$, if the tree is balanced. Finding the first interval before/after a query point or interval can also be found in $O(\log n)$. Finally, given a node in the tree, the next node in the ordering can be found in $O(\log n)$. Cormen et al. [10] provide a thorough introduction to interval trees, and how they can be

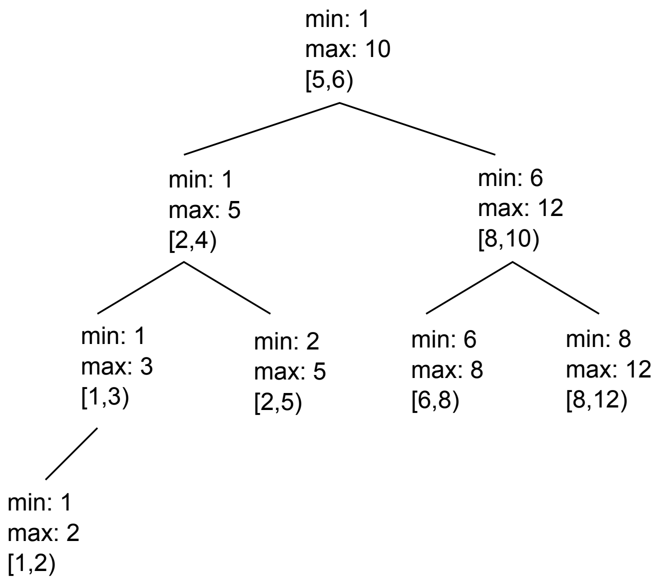


Figure 4: Example of an interval tree holding the intervals (in order) $[1,2)$, $[1,3)$, $[2,4)$, $[2,5)$, $[5,6)$, $[6,8)$, $[8,10)$, and $[8,12)$. If the tree is balanced (which is the case), retrieval and query operations can be done in $O(\log n)$. For example, finding the interval that contains 7 can be found by visiting the root node $[5,6)$, then $[8,10)$ and then finally $[6,8)$. The min and max values of every node, along with the stored interval, allow to quickly find the direction to visit nodes or determine that no overlapping node is present.

implemented through augmenting a Red Black tree, a self balancing binary tree data structure.

For both ORs and resources, interval trees are used to store all relevant intervals (surgical case start + duration for ORs; surgical phase start + duration for resources), where each node also stores a pointer to the relevant surgical case. The main motivation for using interval trees to maintain OR or resource schedules is that they allow to store intervals with any precision of time (e.g. seconds, but any unit is theoretically possible) while still ensuring fast, i.e. $O(\log n)$, retrieval properties and having a low memory footprint (only storing start/endpoints per interval + some pointers).

The schedules of OR $o \in O$ are denoted $T_{o,d}$ for each day of the planning horizon $d \in D$. The schedules of resource $r \in R$ are denoted $T_{r,d}$. Similarly, the availability intervals of an OR o/resource r are stored in interval trees $AV_{o,d}/AV_{r,d}$ for each day d.

2. **Variable definitions:** Let S' denote the set of all scheduled surgical cases up to this point. Let $S'_{o,d}$ denote the subset of S' that were scheduled in OR o, on day d. Let $S'_{r,d}$ denote the subset of S' that use resource r on day d. Let $T(s)$ denote the scheduled start time of surgical case s. Let $T_r(s)$ denote the scheduled start time of a surgical phase for assigned resource $r \in AR$.

3. **Schedule cases:** For each tuple (s, o, d, AR) in the priority feasible list:

(a) Find the earliest starting time t for case s as:

$$t = \max_{\substack{s' \in S'_{o,d}: \\ p_{s'} < p_s}} (T(s') + d_{s'})$$

i.e. the earliest starting time is after the end time of the last surgical case of a lower priority case in the same room on the same day.

(b) Check if $[t, t + d_s)$ is contained in an interval in $AV_{o,d}$. If not, find the next interval $[u, v)$ in $AV_{o,d}$ and $t := u$. If no such interval $[u, v)$ exists, there is no more availability of room o on day d; leave s unscheduled and go to 3.
Note: *To accommodate block scheduling, this check should only consider availability intervals in $AV_{o,d}$ that correspond to the surgical discipline of s.*

(c) Check if $[t, t + d_s)$ overlaps with any interval $[T(s'), T(s') + d_{s'}) \in T_{o,d}$. If yes, $t := T(s') + d_{s'}$ and go to 3b.

(d) Check if $[t + s_{sj}, t + s_{sj} + d_{sj})$ is contained in an interval in $AV_{r,d}$, for each $(j, r) \in AR$. If a resource r is unavailable, find the next interval $[u, v)$ in $AV_{r,d}$, let $t := u - s_{sj}$ and go to 3b. If no such interval $[u, v)$ exists, there is no more availability for resource r on day d; leave s unscheduled and go to 3.

(e) Check if $[t + s_{sj}, t + s_{sj} + d_{sj})$ overlaps with any interval $[T_r(s'), T_r(s') + d_{s'j}) \in T_{r,d}$ for any $(j, r) \in AR$. If yes, $t := T(s') + d_{s'j} - s_{sj}$ and goto 3b.

(f) Schedule surgical case: $S' := S' \cup \{s\}$, $T(s) := t$, Insert($[T(s), T(s) + d_s), T_{o,d}$), and Insert($[T(s) + s_{js}, T(s) + s_{js} + d_{js}), T_{r,d}$) for each $(j, r) \in AR$.

At the end of this procedure, the result is a feasible schedule for all $s \in S'$, while $s \in S \backslash S'$ are left unscheduled.
Essentially, the algorithm just described constructs a schedule in the order defined by the priority feasible list. At each step, the algorithm maintains a schedule for each OR/resource that holds when it is available or occupied. The main loop defined in step 3 scans these schedules in order to find the earliest time at which the current surgical case s can be inserted, respecting its assigned OT and resources. If no such insertion position is found on day d, due to lack of available time, the surgical case s is left unscheduled.

3.2 Schedule evaluation

Given a priority feasible list l, its corresponding schedule can be computed using the list decoding procedure described in Section 3.1 (denoted GenerateSchedule(l)):

$$(T(s), S', T_{o,d}, T_{r,d}, S'_{o,d}, S'_{r,d}) := \text{GenerateSchedule}(l)$$

with $T(s)$ containing the scheduled start time of each case s; S' the set of scheduled cases; $S'_{o,d}, S'_{r,d}$ the set of scheduled cases for OR o, resource r on day d; and $T_{o,d}$, $T_{r,d}$ containing the occupied intervals of each OR o/resource r.
The schedule can be evaluated with respect to the following objectives:

1. Total duration of surgical cases left unscheduled $= \sum_{s \in S \backslash S'} d_s$

2. Number of scheduled OR days $= \sum_{d \in D} \sum_{o \in O} (T_{o,d} \neq \emptyset)$
   where $(T_{o,d} \neq \emptyset) = \begin{cases} 1 & \text{If } |T_{o,d}| > 0 \\ 0 & \text{otherwise.} \end{cases}$

3. Number of surgical cases scheduled in 'if-necessary' room $= |IfNec|$ with:

$$IfNec = \{(s, o, d, AR) \in l | o \in O_s^3 \text{ and } s \in S'\}$$

4. Number of resource 'overloads':

$$Overload = \sum_{d \in D} \sum_{r \in R} \max(|O_{r,d}| - C_r, 0)$$

   with:

$$O_{r,d} = \{o \in O | \exists (s, o, d, AR) \in l \text{ and } s \in S'_{r,d}\}$$

   denoting the distinct ORs in which a resource r is scheduled on day d.

5. Number of resource transfers for resource r on day d: let $l^{S'}_{r',d'}$ denote the *sub*list of l containing (s, o, d, AR) for which $s \in S'$ (it is scheduled), $d = d'$ (assigned to day d') and $\exists (j, r') \in AR$ (has r' assigned to it).

   Then the number of transfers for resource r on day d can be determined by considering all adjacent pairs (s_k, o_k, d_k, AR_k) ,(s_l, o_l, d_l, AR_l) in $l^{S'}_{r,d}$, and checking if $o_k \neq o_l$.

6. Total resource affinity cost:

$$TotalResourceAffinityCost = \sum_{s \in S'} \sum_{(j_1, r_1),(j_2, r_2) \in AR} AF(r_1, r_2)$$

7. Number of surgical cases scheduled in preferred room $= |Pref|$ with:

$$Pref = \{(s, o, d, AR) \in l | o \in O_s^1 \text{ and } s \in S'\}$$

8. Total OR idle time:

$$TotalORIdleTime = \sum_{d \in D} \sum_{o \in O} \left(Max(T_{o,d}) - Min(T_{o,d}) - \sum_{s \in S'_{o,d}} d_s \right)$$

   with $Min(T_{o,d})$,$Max(T_{o,d})$ denoting the first start (last end) time of a surgical case in room o on day d.

9. Total resource idle time:

$$TotalResourceIdleTime = \sum_{d \in D} \sum_{r \in R^{IDLE}} \Big(Max(T_{r,d}) - Min(T_{r,d}) - \sum_{(T(s)+s_{sj}, T(s)+s_{sj}+d_{sj}) \in T_{r,d}} d_{sj} \Big)$$

   with $Min(T_{o,d})$,$Max(T_{o,d})$ denoting the first start (last end) time of a surgical case in room o on day d.

3.3 Local search procedure

The schedule constructed by the list decoding procedure may be arbitrarily bad. Complex resource dependencies may not be resolved due to the sequence of the cases in the priority feasible list, or a particular resource may be assigned to too many surgical cases. Therefore, the list decoding procedure is used within a local search framework to find a list sequence resulting in a good quality schedule. The local search modifies both the sequence of the surgical cases in the priority feasible list, as well as the assigned resources, OR or operating day of the surgical cases in order to find better quality schedules.

The following local search neighbourhoods have been developed that manipulate a given list l to obtain a new list l':

- Shift (**S**): shift a tuple $(s, o, d, AR) \in l$ to a new position, maintaining the priority feasible nature of the list.
- Change Day (**CD**): given a tuple $(s, o, d, AR) \in l$, replace d by $d' \in D_s$.
- Change OR (**COR**): given a tuple $(s, o, d, AR) \in l$, replace o by $o' \in O_s$.
- Change Assigned Resources (**CAR**): given a tuple $(s, o, d, AR) \in l$, with $AR = \{(j, r) | j \in RT_s^r, r \in R_j\}$, select a required resource (j, r) and a resource $r' \in R_j$ and replace (j, r) by (j, r').

A random improving-or-equal local search algorithm has been developed, using these neighbourhoods to improve an initial feasible priority list and corresponding schedule. Pseudocode of this procedure is presented in Algorithm 1. Note that the *lexicographical* improving-or-equal comparison (denoted using operator $\preceq$) is based on the ordering defined earlier. The algorithm requires as input an initial feasible

Algorithm 1 Local search procedure

```
Require: $f : l \mapsto \mathbb{R}$                    ▷ $f$ applies list decoding and evaluation to $l$
Require: $l$                                           ▷ Initial priority feasible list
  while termination criterion not met do
    $N \leftarrow$ SelectNeighbourhood(S,CD,COR,CAR)
    $l' \leftarrow N(l)$                               ▷ Obtain a neighbouring solution from $l$
    if $f(l') \preceq f(l)$ then      ▷ Only accept lexicographic improving/equal solutions
      $l \leftarrow l'$
    end if
  end while
  return $l$
```

priority list from which the local search is started. We have opted to construct this initial list randomly, relying on the local search algorithm to find a new list that results in a good quality schedule. Pseudocode of this method that constructs the initial random list is presented in Algorithm 2.

3.4 Optional resource assignment

After the local search phase has finished, the final schedule is constructed. Up to this point, only required resource dependencies have been considered, while optional

Algorithm 2 Procedure to construct random initial feasible list

```
$l \leftarrow <>$                                    ▷ $l$ is initially an empty sequence
PrioritySort($S$)                    ▷ Assume $S$ is sorted by increasing priority number
for $s \in S$ do             ▷ Iterate over surgical cases in increasing priory number
    $o \leftarrow$ SelectRandomly($O_s$)
    $d \leftarrow$ SelectRandomly($D_s$)
    $AR \leftarrow \emptyset$                      ▷ Will contain (resource type, resource) tuples
    $R^{ASSIGNED} \leftarrow \emptyset$            ▷ Will contain already assigned resources for case $s$
    for $j \in RT_s^r$ do
        for $i = 1 \dots C_{sj}$ do
            $r_i \leftarrow$ SelectRandomly($R_j \backslash R^{ASSIGNED}$)
            $AR \leftarrow AR \cup (j, r_i)$
            $R^{ASSIGNED} \leftarrow R^{ASSIGNED} \cup r_i$
        end for
    end for
    $l \leftarrow l \cup (s, o, d, AR)$                                ▷ Append tuple to list
end for
return $l$
```

resource dependencies have been left unassigned. Essentially, optional resource dependencies could have been handled in a similar fashion as required dependencies. Optional resources can be assigned to a surgical case s, be scheduled by the list decoding procedure, and be manipulated by the local search. For this, the list decoding procedure should be suitably modified to not prohibit a case from being scheduled due to unavailability of an optional resource.
However, treating the optional resources in the same manner as required resources would slow down the approach unacceptably in hospitals where optional resources are numerous.
A two-phase approach has been developed to cope with this problem, performing the optional resource assignment after the surgical case schedule has been constructed. The optional resource assignment is implemented using a greedy approach: it considers the scheduled surgical cases one by one, and assigns the resource that minimally increases the objective function. Note that in the first phase, the number of optional resources left unassigned is irrelevant. Due to the two-phase approach, this value does not change in the first phase (and will be equal to the total number of optional resources required by the surgical cases, as none will be assigned). However, in the second phase, when the optional resources are assigned, the number of optional resources left unassigned is lexicographically evaluated as fourth most important objective (as described in Section 2.2.4).

4 Modelling examples

The flexibility of the resource dependencies is illustrated by modelling two common practical considerations in surgical case scheduling, using generalized resource dependencies.

4.1 Post-anaesthetic care unit (PACU) consideration

The availability of a bed in the post-anaesthetic care unit (PACU) is an important consideration for surgical case schedules. Generally, patients undergo surgery under local, regional or general anaesthesia, and require post-operative monitoring (typically a few hours) in the PACU to assess recovery thereof. A bed in the PACU must be available when surgery ends to ensure proper monitoring. If no beds are available for some time, surgeries may be delayed or even postponed to a later date. The following shows how such a dependency can be modelled using the generalized resource constraints.

- Define a resource type j_{PACU} denoting the PACU. Define as many resources r as there are beds in the PACU, with $RT_r := \{j_{PACU}\}$.
- For each surgical case s (assuming all surgeries require post-anaesthetic care), add a required resource dependency on j_{PACU}, i.e. $RT_s^r := RT_s^r \cup \{j_{PACU}\}$ with $C_{sj_{PACU}} = 1$ and $s_{sj_{PACU}} = d_s$, $d_{sj_{PACU}} =$ Required monitoring time in the $PACU$. Thus, a required resource dependency is defined with surgical phase starting *after* the surgical case *ends*. Therefore, the surgical case can only be scheduled at a time when bed availability in the PACU can be ensured after the surgery ends.

4.2 Instrument kits

Another common practical consideration is the usage of *instrument kits*, standard sets of surgical tools (scalpels, scissors, clamps). Clearly such tools are necessary during surgery and thus need to be available. It is important to account for the fact that these also must be cleaned and sterilized after each surgery. Cleaning is typically performed in a dedicated facility, which may take significant time. The *turnaround* time of this cleaning process can therefore not be neglected. When instrument kits are in limited supply (some kits are specialized for example), this may require extra attention during scheduling.
Again such a dependency can be modelled:

- Define resource types $j_{instr1}, j_{instr2}, \ldots, j_{instrK}$ as much as there are different kinds of instrument kits (e.g. K). Define as many resources r as there are instrument kits of each type, with $RT_r := \{j_{instr1}, j_{instr2}, \ldots\}$. Large kits may serve multiple purposes thus it is possible that $|RT_r| > 1$.
- Add resource dependencies to surgical cases s, accommodating their instrument kit dependencies, e.g. $RT_s^r := RT_s^r \cup \{j_{instr1}\}$ and accordingly s_{sj}, d_{sj}, with d_{sj} sufficiently long to account for cleaning turnaround time.

5 Computational experiments

5.1 Experimental setup

Data for the surgical case scheduling problem was provided by Dotnext, as part of the research project. The algorithm has been implemented in Java 1.8 and provides an XML file based interface for reading problem instances and writing the generated

$\lvert D\rvert$	# instances	Avg. $\lvert R\rvert$	Avg. $\lvert S\rvert$
1	11	276.1	86.5
2	9	285.2	175.2
3	7	288.6	258.4
4	11	288.8	247.9
5	8	292.1	306.5
6	4	293.5	339.5
7	2	293.5	429.5

Table 4: General problem instance characteristics. Instances are grouped by their planning horizon, ranging from 1 to 7 days.

schedules. The QCare OR application interfaces with the algorithm by means of writing and reading XML files and displaying the results in the user interface.
A dataset of 52 problem instances was obtained from a Belgian hospital that uses QCare OR for managing the OT. This hospital has an OT consisting of 24 ORs, of which 18 are general purpose ORs and 6 are specialized.
General characteristics of this dataset can be found in Table 4. We distinguish the problem instances by their time horizon, which ranges from 1 to 7 days. The horizon relates to problem size. Single-day instances have fewer appointments to be scheduled than multi-day instances. As can be seen in Table 4, the number of resources under consideration is quite high. In general, the surgical cases have only one required resource type dependency, a surgeon, and 2-3 optional resource dependencies, representing the remaining surgical team (anaesthesiologists, nursing staff). The specified surgical phase $[s_{sj}, d_{sj})$ for surgeons is smaller than the surgical case interval $[0, d_s)$, i.e. surgeons only need to be present during the incision phase. In addition, surgical cases were always fixed to a specific surgeon. Incorporation in the presented model can be achieved by specifying a unique 'resource type' specific to each surgeon. Finally, no MSS is imposed, therefore all ORs are available for scheduling surgical cases.
All tests reported in the following sections have been performed on a workstation computer equipped with two eight-core Intel Xeon 2650 v2 2.6 GHz processors and 128 GB of main memory (RAM), running a Linux-based operating system. Only one processing thread is used per test, as the algorithm does not employ parallelism. The system was therefore used to perform up to 16 tests in parallel (limiting available memory to 8 GB for each test) to reduce overall computation time.

5.2 Results and discussion

5.2.1 Convergence of the algorithm

One important aspect is whether or not the algorithm converges in a reasonable time, considering that the number of resources (and thus the problem size) is quite large. To analyse this, the algorithm was tested on all instances with different time-outs as termination criterion. The following time-outs were tested: 30 s, 60 s, 120 s, 300 s, 600 s.
Figure 5 shows the relative improvement that can still be found after an initial 30 seconds of running the local search algorithm. The results have been averaged over all instances. They were grouped per planning horizon (ranging from 1 to 7 days) to

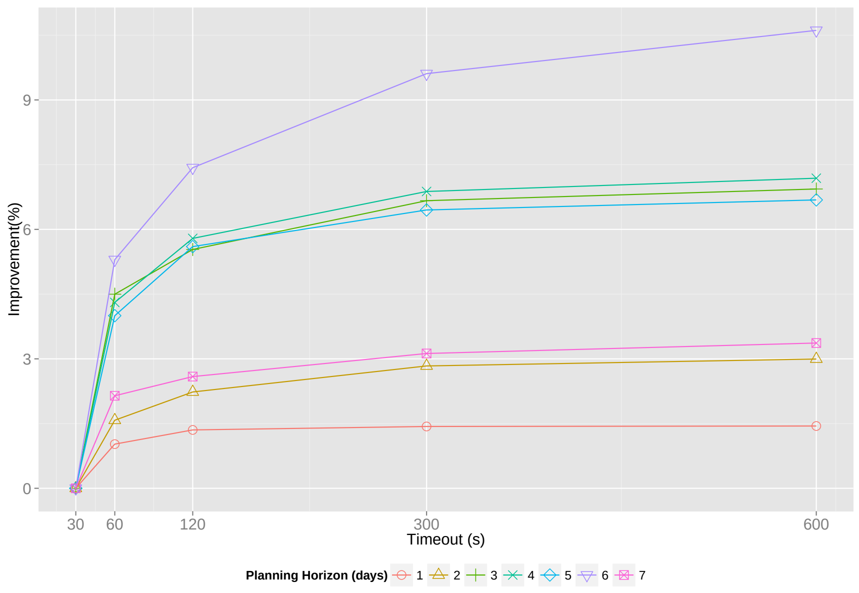


Figure 5: Convergence rate of the algorithm for the different instance subsets (grouped by planning horizon). The graph shows the relative improvement of the local search algorithm that can still be found after an initial 30 second optimization. Note that the results obtained for different planning horizons are incomparable.

show how fast the algorithm converges with respect to the problem size. It is clear that for small instances (1-2 day instances) the algorithm is able to converge in a relatively short time span of 1-2 minutes. The objective of the presented research was to construct schedules for day instances in a reasonable time and to allow for near-interactive use. The analysis reveals that the larger planning horizons (up to 1 week) require additional time (up to 10 minutes) for the improvement process to start to converge. However, in a weekly planning setting, the required time is less of an issue.

5.2.2 Comparison with practice

The quality of the obtained schedules also requires analysis. The generated schedules for the set of 52 instances were compared to the schedules made by manual planners, available in the QCare OR application. An important note is that these schedules rarely were completely *feasible*: overlap could be detected between surgical cases, surgical phases would overlap for some resources, or surgical cases were scheduled outside of availabilities of the ORs/resources. For the purpose of comparison, these surgical cases were considered left unscheduled since the objective here is to construct feasible schedules (respecting all timing information). In the case of overlap between two surgical cases, or their required resources, only one is left unscheduled. Infeasibilities due to overlap for optional resource dependencies were handled by leaving the optional resource unassigned.
We tested the algorithm on all instances with the same time-outs of Section 5.2.1 (30 s, 60 s, 120 s, 300 s, 600 s). Additionally, each test was performed ten times

with a different random seed (ranging from 1 to 10) to reduce random effects. Table 5 reports the results obtained by the heuristic approach on the one-day instances. This subset of instances is of particular importance as it represents a common use case where e.g. the detailed schedule of surgeries is constructed one day before they are performed. The table reports on the different objective function components: the violation of soft constraints and the two primary objectives. It indicates both the average result (over all instances and 10 runs per instance) and the average standard deviation (standard dev. calculated over 10 runs per instance and averaged over all instances). An additional measure is also presented to report on the expected quality of the schedules. The **Room Gap** measure shows the average deviation from a lower bound on the number of OR-days required to assign all surgeries that were scheduled. This lower bound was calculated by solving a bin packing MIP-model (refer to Appendix A), taking into account operating rooms and their daily capacity (which varies across ORs) and the surgery durations.
Clearly, the presented approach is able to schedule more surgical cases than what was obtained by the manual planner, without violating any of the hard constraints. Less than one surgical case is left unscheduled on average, indicating that it was almost always possible to make a completely feasible schedule containing all surgical cases. The two primary objectives, the sum of unscheduled surgery duration and number of OR-days, converge in less than 60 seconds. This small execution time allows for interactive use. Soft constraint violations can be further reduced by allowing additional computation time. The gap from the lower bound regarding required OR-days is on average between 25 and 30%. Although this may still seem large, it is expected that the lower bound is not tight as resource availabilities are not considered when computing the bound. Therefore, we expect that the schedules are rather tight. With respect to the remaining secondary soft constraints, the approach is able to schedule more optional resources (feasibly) and minimize resource affinities. Evidently, this results in higher total resource overload and more resource transfers, as more resources are assigned.
Table 6 reports on the results obtained for the entire set of instances (averaged over all instances, ten runs per instance). The results are quite similar to the ones obtained on the one-day instance subset. Clearly it remains beneficial to increase the time-out of the algorithm to 600 seconds to improve convergence of the algorithm on the larger instances.

6 Conclusion and future work

6.1 Conclusion

This paper has reported on a flexible decision support model for multi-day OT scheduling that encompasses many considerations from hospital practice. This development has been partly supported by Dotnext, a company that commercializes a software suite (QCare OR) for managing the operating theatre.
A surgical case scheduling model with generalized resource dependencies was introduced to cope with a broad variety of scheduling considerations: human dependencies (surgeons, anaesthesiologists, instrumentalists, nurses) as well as material dependencies (e.g. large surgical equipment), dependencies during specific surgical phases, etc. The aim of the model is to assist in scheduling as many surgical cases as possible, within availabilities of all considered ORs and resources, to re-

	Lexicographic approach with timeout T					Manually
Objective	$T = 30$ s	$T = 60$ s	$T = 120$ s	$T = 300$ s	$T = 600$ s	scheduled
	mean (avg. sd.)	mean (avg. sd.)	mean (avg. sd.)	mean (avg. sd.)	mean (avg. sd.)	mean (avg. sd.)
Appointments left unscheduled	**0.7 (0)**	**0.7 (0)**	**0.7 (0)**	**0.7 (0)**	**0.7 (0)**	8.9 (-)
Total duration appointments left unscheduled	**217.4 (5.7)**	**217.4 (5.7)**	**217.4 (5.7)**	**217.4 (5.7)**	**217.4 (5.7)**	1577.2 (-)
OR days	**19.6 (0.7)**	**19.2 (0.7)**	**19.1 (0.7)**	**19 (0.7)**	**19 (0.7)**	23.7 (-)
OR days gap from LB	**30.4 (4.9)**	**28.2 (5)**	**27.2 (4.7)**	**26.9 (4.8)**	**26.9 (4.8)**	110.8 (-)
Total room idle time	**26.9 (25.9)**	**18.4 (15.9)**	**16.7 (16.2)**	**17 (16.3)**	**17 (15.8)**	827.8 (-)
Total resource idle time	**2063.4 (213)**	**2041 (224.8)**	**2033.2 (220.4)**	**2005.6 (216)**	**2009.1 (214.2)**	2596.6 (-)
Optional resources left unscheduled	**75.1 (1.9)**	**74.7 (1.9)**	**74.9 (1.9)**	**74.7 (2.3)**	**74.7 (1.8)**	144.3 (-)
Total resource overload	8.9 (2.6)	8.2 (2.2)	8 (2.3)	8 (2.2)	8.2 (1.9)	1.6 (-)
Total resource transfers	12.7 (2.9)	11.8 (2.4)	11.4 (2.4)	11.4 (2.5)	11.6 (2.4)	3.2 (-)
Total resource affinity cost	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (-)

Table 5: Results of the heuristic approach on instances with a one-day horizon. The results are averages over all instances (with ten runs per instance) of this subset. Performance is reported for different time-outs: 30, 60, 120, 300 and 600 seconds. A comparison to the performance measures on schedules made by manual planners is also provided. Bold highlighting indicates better than practice.

	Lexicographic approach with timeout T					Manually
Objective	$T = 30$ s	$T = 60$ s	$T = 120$ s	$T = 300$ s	$T = 600$ s	scheduled
	mean (avg. sd.)	mean (avg. sd.)	mean (avg. sd.)	mean (avg. sd.)	mean (avg. sd.)	mean (avg. sd.)
Appointments left unscheduled	**13.2 (0.3)**	**13.1 (0.1)**	**13.1 (0.1)**	**13.1 (0.1)**	**13.1 (0)**	23.2 (-)
Total duration appointments left unscheduled	**2013.9 (62.3)**	**1980.7 (26.5)**	**1975.1 (18.5)**	**1970.6 (11.6)**	**1969.7 (9.7)**	4190.2 (-)
OR days	**52.9 (1.4)**	**50.3 (1.2)**	**48.7 (1)**	**47.5 (1)**	**46.9 (1)**	62 (-)
OR days gap from LB	**53.9 (4.8)**	**47 (4.1)**	**43.1 (3.4)**	**40.3 (3.4)**	**39 (3.4)**	112.9 (-)
Total room idle time	**228.4 (113.9)**	**118 (64.1)**	**59.3 (39.7)**	**34.2 (27)**	**26.6 (23)**	2163 (-)
Total resource idle time	**5676.4 (497.3)**	**5116.5 (402.6)**	**4776.8 (371.5)**	**4560.5 (339.7)**	**4480.4 (333.3)**	6749.3 (-)
Optional resources left unscheduled	**185.9 (2.7)**	**185.9 (2.7)**	**185.8 (2.9)**	**185.7 (2.9)**	**185.6 (2.9)**	376.4 (-)
Total resource overload	45.9 (5.5)	32.7 (3.9)	25.5 (3.8)	20.8 (3.2)	19.6 (3)	4.8 (-)
Total resource transfers	55.8 (5.9)	41.3 (4.3)	32.9 (4)	27.4 (3.8)	25.8 (3.6)	8 (-)
Total resource affinity cost	**-6.8 (1)**	**-7.7 (1.1)**	**-8.4 (1.2)**	**-9.1 (1.3)**	**-9.3 (1.2)**	0 (-)

Table 6: Results of the heuristic approach on the entire set of instances. The results are averages over all instances (with ten runs per instance). Performance is reported for different time-outs: 30, 60, 120, 300 and 600 seconds. A comparison to the performance measures on schedules made by manual planners is also provided. Bold highlighting indicates better than practice.

duce the number of ORs that need to be opened for performing surgical cases, to minimize violations of soft constraints and to optimize several measures of resource efficiency. A heuristic approach to this feature-rich model has been developed, scaling favourably with problem size. An important achievement is that the algorithmic complexity of the schedule generation approach does not depend on the scheduling precision and is therefore able to schedule up to any precision of time.
The approach has been validated on a set of problem instances obtained from a Belgian hospital that uses QCare OR for managing the operating theatre. The algorithm has been implemented and has been tested in conjunction with the QCare OR application.
Computational experiments were conducted on a set of 52 problem instances obtained from the hospital. Our results show that the approach is able to schedule more surgical cases than a manual planner in a feasible way (e.g. not violating opening hours of ORs, availability of resources), whilst further decreasing the required number of ORs. In addition, secondary resource performance and efficiency measures are improved.

6.2 Future work

Although the work presented in this paper has had a strong focus on flexibility and generality, there are still elements that were left out of scope.
Most notably is that the approach disregards the uncertain nature of the scheduling problem. In practice, it is only possible to estimate the duration of surgeries; the exact duration depends on the specific case and may vary depending on the surgeon, the patient and possible complications. However, the results presented in this work have indicated that the generated schedules are quite dense. Consequently, variation in surgery durations may cause delays and shifts in the schedule, ultimately resulting in overtime or postponed/cancelled surgeries. Some slack may need to be introduced to cope with this uncertainty. In future work, we therefore plan to examine how the schedules can be made more robust to variation, either by a) simply adding some slack to surgical durations, or b) developing a more advanced model taking into account stochastic surgery durations (with a known distribution from historic data) through *sample average approximation* (SAA). The former approach is clearly easier to implement but may be too conservative/aggressive depending on the surgery duration distribution (which often are modelled with long-tailed distributions such as the log-normal distribution [31]). Therefore the latter approach may be more appropriate in correctly accounting surgical duration variation, at the cost of increased computational effort.
Apart from the uncertain nature of the problem, other considerations from practice were not yet considered in this work. One observation was pointed out by a reviewer: the model does not allow resources to be assigned to multiple, non-overlapping, surgical phases. Such a modelling feature may be interesting, for example resources used at the start and end of a surgery but not in the meantime.

Acknowledgement: We would like to express our thanks to Dotnext, for providing critical insights in OT scheduling practices in hospitals along with relevant data. We further extend our thanks to the reviewers of this work, providing many valuable comments. This work was supported by the Belgian Science Policy Office (BELSPO) in the Interuniversity Attraction Pole COMEX. (http://comex.ulb.ac.be) and funded by both a Ph.D. grant and research project 120604 of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

A Appendix: MIP model for operating room lower bound

Let denote:

$$x_{sod} = \begin{cases} 1 & \text{if surgical case } s \text{ is assigned to OR } o \text{ on day } d, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

$$y_{od} = \begin{cases} 1 & \text{if OR } o \text{ is used on day } d, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Using these decision variables, a binpacking MIP-model can be formulated that provides a lower bound on the number of required OR-days to plan $PLANNED$ (which is a parameter) surgical cases. Given parameter $PLANNED$, solving the MIP-model to optimality (e.g. by using Gurobi) provides the minimal number of OR-days by expression (5).

$$\text{Minimize} \sum_{o \in O} \sum_{d \in D} y_{od} \tag{5}$$

$$\text{Subject to :} \sum_{o \in O_s} \sum_{d \in D_s} x_{sod} \leq 1 \qquad \forall s \in S \tag{6}$$

$$\sum_{\substack{s \in S: \\ d \in D_s}} d_s \cdot x_{sod} \leq y_{od} \cdot \sum_{\substack{[start,end) \in AV_{o,d}: \\ CAP = end - start}} CAP \qquad \forall o \in O, d \in D \tag{7}$$

$$\sum_{s \in S} \sum_{o \in O_s} \sum_{d \in D_s} x_{sod} \geq PLANNED \tag{8}$$

$$\tag{9}$$

$$x_{sod} \in \{0,1\} \qquad \forall s \in S, o \in O_s, d \in D_s \tag{10}$$

$$y_{od} \in \{0,1\} \qquad \forall o \in O, d \in D \tag{11}$$

References

[1] A. Agnetis, A. Coppi, M. Corsini, G. Dellino, C. Meloni, and M. Pranzo. Long term evaluation of operating theater planning policies. *Operations Research for Health Care*, 1(4):95–104, Dec. 2012. doi: 10.1016/j.orhc.2012.10.001.

[2] R. Aringhieri, P. Landa, P. Soriano, E. Tànfani, and A. Testi. A two level metaheuristic for the operating room scheduling and assignment problem. *Computers & Operations Research*, 54:21–34, Feb. 2015. doi: 10.1016/j.cor.2014.08.014.

[3] R. Aringhieri, P. Landa, and E. Tànfani. Assigning surgery cases to operating rooms: A VNS approach for leveling ward beds occupancies. *Electronic Notes in Discrete Mathematics*, 47:173–180, Feb. 2015. doi: 10.1016/j.endm.2014.11.023.

[4] J. Beliën and E. Demeulemeester. Building cyclic master surgery schedules with leveled resulting bed occupancy. *European Journal of Operational Research*, 176(2):1185–1204, Jan. 2007. doi: 10.1016/j.ejor.2005.06.063.

[5] J. Beliën and E. Demeulemeester. A branch-and-price approach for integrating nurse and surgery scheduling. *European Journal of Operational Research*, 189 (3):652–668, Sept. 2008. doi: 10.1016/j.ejor.2006.10.060.

[6] M. E. Bruni, P. Beraldi, and D. Conforti. A stochastic programming approach for operating theatre scheduling under uncertainty. *IMA Journal of Management Mathematics*, Jan. 2014. doi: 10.1093/imaman/dpt027.

[7] B. Cardoen, E. Demeulemeester, and J. Beliën. Sequencing surgical cases in a day-care environment: An exact branch-and-price approach. *Computers & Operations Research*, 36(9):2660–2669, Sept. 2009. doi: 10.1016/j.cor.2008.11.012.

[8] B. Cardoen, E. Demeulemeester, and J. Beliën. Operating room planning and scheduling: A literature review. *European Journal of Operational Research*, 201(3):921–932, 2010. doi: 10.1016/j.ejor.2009.04.011.

[9] B. Cardoen, E. Demeulemeester, and J. Van der Hoeven. On the use of planning models in the operating theatre: results of a survey in Flanders. *The International journal of health planning and management*, 25(4):400–14, 2010. doi: 10.1002/hpm.1027.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*, chapter Interval trees, pages 348–355. MIT Press and McGraw-Hill, 3rd edition, 2009. ISBN 0-262-03384-4.

[11] B. Denton, J. Viapiano, and A. Vogl. Optimization of surgery sequencing and scheduling decisions under uncertainty. *Health Care Management Science*, 10 (1):13–24, Nov. 2006. doi: 10.1007/s10729-006-9005-4.

[12] H. Fei, C. Chu, and N. Meskens. Solving a tactical operating room planning problem by a column-generation-based heuristic procedure with four criteria. *Annals of Operations Research*, 166(1):91–108, Aug. 2008. doi: 10.1007/s10479-008-0413-3.

[13] H. Fei, N. Meskens, and C. Chu. A planning and scheduling problem for an operating theatre using an open scheduling strategy. *Computers & Industrial Engineering*, 58(2):221–230, Mar. 2010. doi: 10.1016/j.cie.2009.02.012.

[14] F. Guerriero and R. Guido. Operational research in the management of the operating theatre: a survey. *Health care management science*, 14(1):89–114, Mar. 2011. doi: 10.1007/s10729-010-9143-6.

[15] A. Guinet and S. Chaabane. Operating theatre planning. *International Journal of Production Economics*, 85(1):69–81, July 2003. doi: 10.1016/S0925-5273(03)00087-2.

[16] E. Hans, G. Wullink, M. van Houdenhoven, and G. Kazemier. Robust surgery loading. *European Journal of Operational Research*, 185(3):1038–1050, Mar. 2008. doi: 10.1016/j.ejor.2006.08.022.

[17] P. J. H. Hulshof, N. Kortbeek, R. J. Boucherie, E. W. Hans, and P. J. M. Bakker. Taxonomic classification of planning decisions in health care: a structured review of the state of the art in OR/MS. *Health Systems*, 1(2):129–175, Dec. 2012. doi: 10.1057/hs.2012.18.

[18] R. L. Jackson. The Business of Surgery. *Health Management Technology*, 23 (7):20–22, 2002.

[19] A. Jebali, A. B. Hadj Alouane, and P. Ladet. Operating rooms scheduling. *International Journal of Production Economics*, 99(1-2):52–62, Jan. 2006. doi: 10.1016/j.ijpe.2004.12.006.

[20] M. Lamiri, X. Xie, and S. Zhang. Column generation approach to operating theater planning with elective and emergency patients. *IIE Transactions*, 40 (9):838–852, July 2008. doi: 10.1080/07408170802165831.

[21] A. Macario, T. S. Vitez, B. Dunn, and T. McDonald. Where Are the Costs in Perioperative Care? *Anesthesiology*, 83:1138–1144, 1995.

[22] J. M. Magerlein and J. B. Martin. Surgical demand scheduling: a review. *Health services research*, 13(4):418–33, Jan. 1978.

[23] I. Marques, M. E. Captivo, and M. Vaz Pato. An integer programming approach to elective surgery scheduling. *OR Spectrum*, 34(2):407–427, Dec. 2011. doi: 10.1007/s00291-011-0279-7.

[24] I. Marques, M. E. Captivo, and M. Vaz Pato. Scheduling elective surgeries in a Portuguese hospital using a genetic heuristic. *Operations Research for Health Care*, 3(2):59–72, June 2014. doi: 10.1016/j.orhc.2013.12.001.

[25] N. Meskens, D. Duvivier, and A. Hanset. Multi-objective operating room scheduling considering desiderata of the surgical team. *Decision Support Systems*, 55(2):650–659, May 2013. doi: 10.1016/j.dss.2012.10.019.

[26] D. Min and Y. Yih. Scheduling elective surgery under uncertainty and downstream capacity constraints. *European Journal of Operational Research*, 206 (3):642–652, Nov. 2010. doi: 10.1016/j.ejor.2010.03.014.

[27] I. Ozkaraham. Allocation of Surgeries to Operating Rooms by Goal Programing. *Journal of Medical Systems*, 24(6):339–378, 2000. doi: 10.1023/A:1005548727003.

[28] D.-N. Pham and A. Klinkert. Surgical case scheduling as a generalized job shop scheduling problem. *European Journal of Operational Research*, 185(3): 1011–1025, Mar. 2008. doi: 10.1016/j.ejor.2006.03.059.

[29] A. Riise and E. K. Burke. Local search for the surgery admission planning problem. *Journal of Heuristics*, 17(4):389–414, Sept. 2011. doi: 10.1007/s10732-010-9139-x.

[30] B. Roland, C. Martinelly, and F. Riane. Operating Theatre Optimization : A Resource-Constrained Based Solving Approach. In *2006 International Conference on Service Systems and Service Management*, pages 443–448. IEEE, Oct. 2006. doi: 10.1109/ICSSSM.2006.320503.

[31] D. P. Strum, J. H. May, and L. G. Vargas. Modeling the uncertainty of surgical procedure times: comparison of log-normal and normal models. *Anesthesiology*, 92(4):1160–7, Apr. 2000.

[32] J. T. van Essen, J. M. Bosch, E. W. Hans, M. van Houdenhoven, and J. L. Hurink. Reducing the number of required beds by rearranging the OR-schedule. *OR Spectrum*, 36(3):585–605, Apr. 2013. doi: 10.1007/s00291-013-0323-x.

[33] C. Van Huele and M. Vanhoucke. Analysis of the integration of the physician rostering problem and the surgery scheduling problem. *Journal of Medical Systems*, 38(6):43, June 2014. doi: 10.1007/s10916-014-0043-z.

[34] B. Vijayakumar, P. J. Parikh, R. Scott, A. Barnes, and J. Gallimore. A dual bin-packing approach to scheduling surgical cases at a publicly-funded hospital. *European Journal of Operational Research*, 224(3):583–591, Feb. 2013. doi: 10.1016/j.ejor.2012.09.010.

[35] Z. Zhao and X. Li. Scheduling elective surgeries with sequence-dependent setup times to multiple operating rooms using constraint programming. *Operations Research for Health Care*, June 2014. doi: 10.1016/j.orhc.2014.05.003. Available online.