Computers & Industrial Engineering 97 (2016) 248–257

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Scheduling operating rooms with consideration of all resources, post anesthesia beds and emergency surgeries

Guillermo Latorre-Núñez [a], Armin Lüer-Villagra [a,d], Vladimir Marianov [a,*], Carlos Obreque [b], Francisco Ramis [b], Liliana Neriz [c]

[a] Department of Electrical Engineering, Pontificia Universidad Católica de Chile, Santiago, Chile
[b] Department of Industrial Engineering, Universidad del Bío-Bío, Concepción, Chile
[c] Department of Management and Information Systems, Universidad de Chile, Santiago, Chile
[d] Department of Engineering Sciences, Universidad Andres Bello, Santiago, Chile

ARTICLE INFO

Article history:
Received 30 July 2014
Received in revised form 28 September 2015
Accepted 10 May 2016
Available online 11 May 2016

Keywords:
Operating room scheduling
Emergency surgeries
Hybrid flow shop
Constraint programming
Genetic algorithm

ABSTRACT

Surgery rooms are among the most expensive resources in hospitals and clinics. Their scheduling is difficult because, in addition to the surgical room itself, each surgery requires a particular combination of human resources, as well as different pieces of equipment and materials. Furthermore, after each surgery, a post-anesthesia bed is required for the patient to recover. Finally, in addition to planned surgeries, the scheduling must be made in such a way as to accommodate the emergency surgeries that may arrive during each day, which must be attended within a limited time. We address the surgery scheduling problem considering simultaneously, for the first time, the operating rooms, the post anesthesia recovery, the resources required by the surgery and the possible arrival of emergency surgeries. We propose an integer linear programming model that allows finding optimal solutions for small size instances, we transform it to use constraint programming, and develop a metaheuristic based on a genetic algorithm and a constructive heuristic, that solves larger size instances. Finally, we present numerical experiments.

1. Introduction

The relevance of the problem of planning and scheduling of operating rooms (ORs) comes from the fact that ORs are among the most expensive and fundamental resources of a hospital (Cardoen, Demeulemeester, & Beliën, 2010; Guerriero & Guido, 2011). Among other consequences, poor scheduling may generate idle time intervals, an excess of overtime, and delays or cancellations of surgeries, that are reflected in excess cost and loss of revenue for the hospital.

The process of scheduling of surgeries in ORs has two main stages: the daily assignment of patients and the appropriate sequencing of the surgeries (Cardoen, Demeulemeester, & Beliën, 2009a; Guerriero & Guido, 2011). The first stage consists of determining the set of patients that should be operated on a particular day over a given planning horizon (Guinet & Chaabane, 2003). Patients that are candidates for surgery are generally selected from patient waiting lists that are generated by the hospital (Persson & Persson, 2009). This selection is based on resource availability, surgery priority, and the patient's waiting time in the waiting list. At this point, a surgeon is assigned to the surgery. In the second stage, the daily programming of each operating room is prepared, which is known as the Surgical Table Programming or Operating Table (OT). This activity determines the sequencing of the surgeries in the ORs and is scheduled 24–48 h before the event (May, Spangler, Strum, & Vargas, 2011). In this paper, we concentrate on the second step of the process of surgery scheduling, that is the sequencing of surgeries in the ORs.

The OT programmer should build the sequence taking into account the ORs availability as well as the physical and human resources needed to perform each surgery. At the same time, the programmer must consider the availability of a post-anesthesia recovery bed for the patient, thus making sure that the surgical room is not blocked, which would impact the following surgery. Not considering any of these physical or human resources may lead to the cancellation of scheduled surgeries.

Normally, the OT programmer is a physician or a nurse, familiar with the process of each of the scheduled surgeries. Nevertheless, even though the programmer is familiar with the elements, time and care needed to perform the surgery, achieving a good schedule is a very difficult process when a large number of factors need to be simultaneously addressed. In addition, every day, a number of

* Corresponding author.
E-mail address: marianov@ing.puc.cl (V. Marianov).

http://dx.doi.org/10.1016/j.cie.2016.05.016
0360-8352/

emergency surgeries arrive to the hospital which must be handled as soon as possible. Thus, even when surgeons are very skilled in performing surgery, large amounts of idle time occur due mainly to poor scheduling.

Our main contribution is the construction of the OT simultaneously considering for the first time the operating rooms, resources, post-anesthesia recovery beds, and emergency surgeries. We propose different tools to solve the problem and assess the results using computational tests.

The remainder of the article is organized as follows: Section 2 reviews the literature. Section 3 describes the problem. Section 4 presents the exact solution approaches and the tools proposed to solve of the problem. Section 5 describes the heuristic. Section 6 presents and describes the computer experiments. Finally, in Section 7 we offer conclusions and outline directions for future work.

2. Literature review

The scheduling of ORs is a broad topic. A detailed review can be found in Cardoen et al. (2010) and Guerriero and Guido (2011). Here, we focus on the literature that is directly related. Among the relevant papers are Jebali, Hadj Alouane, and Ladet (2006) and Dekhici and Belkadi (2010), in which the assignment and sequencing of surgeries are treated as a hybrid flow shop (HFS), without considering any of the required additional resources in the scheduling process. Roland, di Martinelly, and Riane (2006) and Roland, Di Martinelly, Riane, and Pochet (2010) emphasize the importance of considering additional resources, specially human, in the surgery scheduling process but the post-anesthesia recovery stage is not taken into account.

In Pham and Klinkert (2008), each surgical intervention is described as a predetermined sequence of activities with a maximum allowed waiting time between two consecutive activities, with a set of resources being assigned to each of the activities. The authors develop a mixed linear integer programming (MILP) model. Vijayakumar, Parikh, Scott, Barnes, and Gallimore (2013) consider the resources in the problem and use a dual bin-packing problem model. Given the number of days and resources available for the schedule, they maximize the number of surgeries to schedule. They model the problem as a MILP and develop a heuristic to solve the problem without considering the second stage (recovery). The surgical working affinity group (group of surgeons, nurses and personnel that usually work together) is included as a limitation in Meskens, Duvivier, and Hanset (2013), and constraint programming (CP) is used to solve the problem. Lee and Yih (2014) determine the start times of the surgeries in the operating room based on the availability of beds in the unit Post-Anesthesia Care Unit (PACU) and the uncertainty of the duration of tasks. The problem is approached as a flexible job shop with fuzzy times. To solve it, they develop a genetic algorithm that determines the order of the surgeries and a heuristic to determine the start times.

Emergency surgeries cannot be programmed in advance, since they may arrive at any moment of the day. However, they must be attended promptly. To address this type of uncertainty, Erdem, Qu, and Shi (2012) formulate a MILP model and a genetic algorithm that allows rescheduling elective surgeries. The proposed model minimizes the cost of postponing surgeries, the overtime, and the cost of rejecting an emergency patient by deriving her to a different hospital. Wullink et al. (2007) use simulation to evaluate reservation of time for emergency surgeries. They evaluate two traditional methodologies: (a) concentrating all reserve capacity into an ORs dedicated exclusively to emergency surgery, and (b) uniformly reserving capacity in all ORs dedicated to elective surgery. The performance measures are waiting time, overtime, and OR utilization. They conclude that the second approach is the most efficient. van Essen, Hans, Hurink, and Oversberg (2012) study how to minimize the emergency surgery waiting time, for which they introduce two new concepts: the "break-in-moment" (BIM) that corresponds to the exact moment in which an emergency surgery can be performed after the conclusion of an elective surgery, and the "break-in-interval" (BII), defined as the interval between two consecutive BIMs. What they look for is distributing BIMs uniformly throughout the day, for which they minimize the maximum BII. The authors assume that surgeries are programmed without dead times in between. In all these works, they use the fact, recognized in the literature, that there exists a so-called 'golden hour': a one-hour period within which care must be provided to trauma emergency patients (Fleet & Poitras, 2011; Newgard et al., 2010).

We synthesize the main literature in the following Table 1, summarizing the related works. The first column lists the authors, while the second to seventh columns show the features of the studied problem. As Table 1 shows, most of the papers partially address the features involved in the problem. Our model covers most of the features: programming for minimum makespan and consideration of the waiting time of emergency surgeries, sequencing and individual assignment of all the medical staff as well as all the resources, and consideration of the two important stages: the OR and the recovery bed.

3. Problem description

A patient's surgery process consists of two main stages. The first stage makes use of the surgery room and it is divided into the OR preparation for surgery, the surgery itself (surgical act), and the cleaning of the OR. The second stage is the post-anesthesia recovery of the patient that takes place in a recovery room.

In general, ORs have different sizes and characteristics, thus a surgery can only be assigned to an operating room the meets the conditions required to perform the operation. Priorities among surgeries may also occur. For example, it may be necessary to move towards the end of the day those surgeries that may contaminate the room to a larger degree.

Each surgery requires a previously assigned physician or surgeon. The surgeon is only present during the surgery act and requires a certain amount of time between interventions (mainly for cleaning, change of clothes, and rest) that depends on the duration of the previously performed surgery.

During a surgery additional resources are required, often scarce. These resources may be nurses, anesthesiologists and other professionals, or physical resources (instruments, imaging equipment, etc.). One or more units of a particular type of resource (human or physical) may be available. Each resource unit assigned to a surgery must be available throughout the preparation and performance of the surgery and has a fixed preparation time that depends on the resource type.

Once the surgery is finished, the patient must be transported to a bed of PACU. Some patients, after surgery, require special care and must be transported to an Intensive Care Unit (ICU). Thus, post-anesthesia recovery may require a special bed.

In addition, demand for emergency surgeries may occur at any time and must be attended within a maximum waiting time.

The objective of the problem is to minimize the closing time of the last operating room in use (makespan).

4. Proposed exact approaches

The problem of scheduling operating rooms, resources and post anesthesia care units can be seen as a hybrid flow shop (HFS) scheduling problem. This is an NP-Hard combinatorial problem

Table 1
Summary of the main literature on OR scheduling.

Research	Type of problem	Medical staff	Resources type	Stochastic aspects	Solution method	Criteria
Addis, Carello, Grosso, and Tànfani (2015)	1–2	–	–	1	1	10
Augusto, Xie, and Perdomo (2010)	1	–	1–5	–	7	2
Cardoen et al. (2009a)	1	1	1–4–5	–	1–4	3–4–5–10
Cardoen, Demeulemeester, and Beliën (2009b)	1	1	1–4–5	–	1–8–9	3–4–5–10
Dekhici and Belkadi (2010)	1	–	1	–	1–5	1
Dios, Molina-Pariente, Fernandez-Viagas, Andrade-Pineda, and Framinan (2015)	3	1	–	–	1–4	10
Erdem et al. (2012)	2	1–3	1–3	2	1–3	3–6–10
Fei, Chu, and Meskens (2009)	3	1	–	–	1–4	3–6–7
Fei, Meskens, and Chu (2010)	1–3	1	1	–	3–4	3–6–7–9
Guinet and Chaabane (2003)	1–3	1	1–3	–	1–4	3–6–10
Guo, Wu, Li, and Rong (2014)	–	2	–	1	1	6–10
Jebali et al. (2006)	1–3	1	1–3	–	1	3–6–10
Lee and Yih (2014)	1	–	1	1	3–4	2–9–10
Marcon and Dexter (2006)	1	1	1	1	4–6	7–10
Marcon et al. (2003)	1	1	1–5	–	6	4–10
Marques, Captivo, and Vaz Pato (2012)	1–3	1	–	–	1	10
Marques, Captivo, and Vaz Pato (2014)	1–3	1	–	–	1–3	10
Meskens et al. (2013)	1	1–2–3	1–4	–	2	1–3–10
Paoletti and Marty (2007)	–	3	–	1	6	10
Perdomo, Augusto, and Xiaolan (2006)	1	–	1–5	–	1	2
Persson and Persson (2009)	1–3	1	1	–	1–6	3–6–10
Pham and Klinkert (2008)	1–2	1–2–3	1–2–3	–	1–7	1
Roland et al. (2006)	1–3	1–2–3	5	–	1–3	3–8
Roland et al. (2010)	1–3	1–2–3	5	–	1–3	3–8
van Essen, Hans, et al. (2012)	1	–	–	2	1–4–6	10
van Essen, Hurink, Hartholt, and van den Akker (2012)	2	1–2–3	1–3	2–3	1	10
Vijayakumar et al. (2013)	1–3	1–2	3	–	1–4	10
Wullink et al. (2007)	–	–	–	2	6	3–7–10
Our model	1	1–2–3	1–3–4–5	1–2	1–2–5	1

Type of problem*:** 1 (scheduling); 2 (rescheduling); 3 (planning). **// *Medical staff*:** 1 (surgeons); 2 (nurses); 3 (anesthesiologists). **// *Resource type: 1 (recovery beds); 2 (ICU); 3 (equipment); 4 (instruments); 5 (others) **// *Stochastic aspects:*** 1 (surgery duration); 2 (emergency surgeries); 3 (surgery cancellations). **// *Solution method:*** 1 (Mathematical programming model); 2 (CP); 3 (GA); 4 (other heuristics); 5 (metaheuristics); 6 (simulation); 7 (Lagrangian relaxation); 8 (Dynamic programming); 9 (Branch-and-price). **// *Criteria:*** 1 (makespan); 2 (total completion time); 3 (overtime); 4 (number of beds); 5 (start time); 6 (cost); 7 (OR utilization); 8 (opening OR); 9 (idle time); 10 (others).

(Gupta, 1988). To learn more about the HFS see Ruiz and Vazquez-Rodriguez (2010) and Ribas, Leisten, and Framinan (2010), as well as the references therein.

A *simple flow shop* is an environment in which products, tasks, or jobs, require processing involving more than one stage. Each stage consists of a machine, and adds to the final result. A *hybrid flow shop* has more than one (parallel) machine in at least one of the stages, and all machines in a stage can perform either the same task, or have different capabilities. Each machine can handle only one product, task or job at a time, and each task is processed by only one machine in each stage.

Our first approach is addressing the problem as a two-stage HFS. Each job (surgery) is assigned to exactly one machine, belonging to the set of parallel machines in stage 1 (ORs), and exactly to one machine belonging to the set of parallel machines at stage 2 (beds of PACU). In both stages, there is a subset of machines that can perform the job. Each job requires additional resources to be processed. There are setup times that are sequence-dependent, and a restriction of 'no wait' to move from stage 1 to stage 2. The objective is to minimize the makespan in stage 1.

Finally, there are urgent jobs (emergency surgeries) that can arrive, which must be attended within a standard amount of time. To address the emergency surgery issue, based on the results of Wullink et al. (2007), we do not consider assigning special ORs for such surgeries. Rather, we use the approach proposed in van Essen, Hans, et al. (2012), consisting in minimizing the maximum waiting time for an emergency surgery, and we transform it into a problem constraint, whereas we limit the maximum waiting time to a pre-determined value (60 min based on the 'golden rule').

Fig. 1 presents an example in which 3 surgeries (a, b, and c) are scheduled in 2 ORs and 2 beds of PACU. Note that the start of surgery c is delayed at stage 1, to have bed availability in stage 2 without a waiting period. The surgeon assigned to surgery c is only present during the surgical act, while the resource used (a nurse, for example) is present since the preparation of the surgery starts until it finishes. If emergency surgeries are required, they can be performed at the beginning of the day, between surgeries or at the end of the day. The double arrow indicates the maximum waiting time for an emergency surgery (one hour). Note that this limit is not satisfied after surgery c starts. If an emergency surgery appears, once c has begun, it must wait until total completion of c. The completion of the cleaning after surgery b in OR 1 determines the makespan.

To solve the problem, we formulate a MILP. We transform it into a Constraint Programming model, as this methodology has shown to produce good quality feasible solutions with low computational cost.

4.1. Mathematical milp model

The notation for the model is as follows.

Indexes	
i, j	surgeries
k	ORs and beds of PACU
h	stages
a	surgeons
r	resources
Parameters	
n	total number of surgeries scheduled

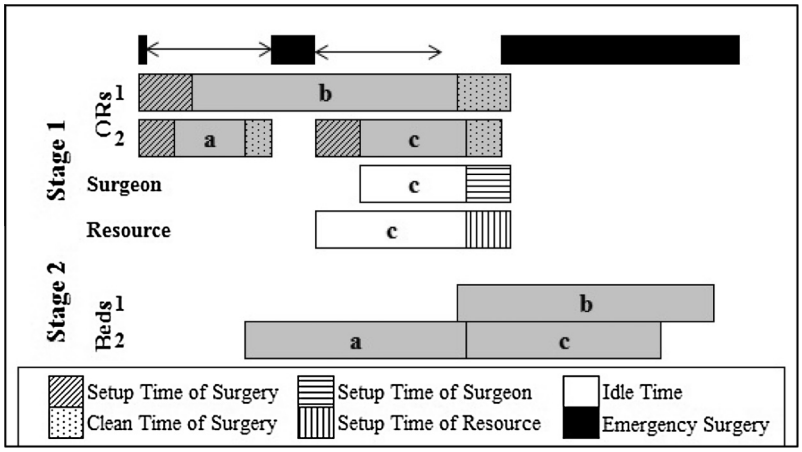

Fig. 1. Scheduling of surgeries.

P_j^h	duration time ($h = 1$ is surgical act, $h = 2$ is post anesthesia recovery) of surgery j
S_j	setup time of surgery j
L_j	clean time of surgery j
S_j^a	setup time of surgeon a, after to performing the surgery j
S_r	setup time of resource r
E_{max}	maximum waiting time for emergency surgery
m^h	numbers of ORs/beds ($h = 1$ is ORs, $h = 2$ is beds)
u_r	numbers of resource r
G	large integer number
Sets	
J	set of surgeries
K_j^h	set of ORs/beds ($h = 1$ is OR, $h = 2$ is bed) where can be assigned to the surgery j
J_k^h	set of surgeries that can be assigned to the OR/bed k ($h = 1$ is OR, $h = 2$ is bed)
M	set of surgeons
M_a	set of surgeries assigned to the surgeon a
U	set of resources
U_j	set of resources required for surgery j
J_r	set of surgeries that require the resource r
B	set of surgeries that require priority
B_j	set of surgeries that cannot precede surgery j

The following are the variables of the model. These variables specify the allocation of surgeries to operating rooms in stage 1 and post-anesthesia beds in stage 2. Also, they allow allocating resources and scheduling in such a way that the maximum waiting time for emergency surgeries, E_{max} is not exceeded.

$x_{jk}^h = 1$ if surgery j, in stage h ($h = 1$ is OR, $h = 2$ is bed), is assigned to the OR or bed k, 0 otherwise.
$y_{ij}^h = 1$ if the end of surgery i precedes the end of surgery j in stage h, 0 otherwise.
$w_{ijr}^1 = 1$ if the start of surgery i precedes the end of surgery j and both surgeries require of resource r, 0 otherwise.
$w_{ijr}^2 = 1$ if surgery i and surgery j are performed in parallel at an instant of time and both surgeries require of resource r, 0 otherwise.
$z_{ij}^1 = 1$ if the gap between the start of surgery i and the end of surgery j is greater than E_{max}, 0 otherwise.
$z_{ij}^2 = 1$ if the completion of the cleaning of surgery i precedes the completion of the cleaning of surgery j, 0 otherwise.
$z_{ij}^3 = 1$ if the parallel performing time of the surgeries i and j is greater than E_{max}, 0 otherwise.
C_i^h = completion time of scheduled surgery i in stage h.
C_{max} = closing time of the last OR.

The formulation of the model is as follows:

$$Min\ C_{max} \tag{1}$$

Subject to :

$$\sum_{k \in K_j^h} x_{jk}^h = 1, \quad \forall h = 1,2; j \in J \tag{2}$$

$$C_i^h \leqslant C_j^h + G^*(1 - y_{ij}^h), \quad \forall h = 1,2; j \in J; i \in J\, i \neq j \tag{3}$$

$$y_{ij}^h + y_{ji}^h = 1, \quad \forall h = 1,2; j \in J; i \in J\, i \neq j \tag{4}$$

$$C_j^1 \geqslant S_j + P_j^1, \quad \forall j \in J \tag{5}$$

$$C_j^1 - C_i^1 \geqslant L_i + S_j + P_j^1 + G^*(x_{ik}^1 + x_{jk}^1 + y_{ij}^1 - 3), \quad \forall j \in J; k \in K_j^1; i \in J_k^1\, i \neq j \tag{6}$$

$$C_j^2 - C_i^2 \geqslant P_j^2 + G^*(x_{ik}^2 + x_{jk}^2 + y_{ij}^2 - 3), \quad \forall j \in J; k \in K_j^2; i \in J_k^2\, i \neq j \tag{7}$$

$$C_j^2 = C_j^1 + P_j^2, \quad \forall j \in J \tag{8}$$

$$x_{ik}^1 + x_{jk}^1 + y_{ij}^1 \leqslant 2, \quad \forall k = 1 \ldots m^1; j \in J_k^1 \cap B; i \in B_j \cap J_k^1 \tag{9}$$

$$C_i^1 - P_i^1 \geqslant C_j^1 + S_j^a - G^* y_{ij}^1, \quad \forall a \in M; j \in M_a; i \in M_a\, i \neq j \tag{10}$$

$$C_i^1 - P_i^1 - S_i \geqslant C_j^1 + S_r - G^* w_{jir}^1, \quad \forall r \in U; j \in J_r; i \in J_r\, i \neq j \tag{11}$$

$$y_{ij}^1 + w_{jir}^1 + w_{ijr}^1 \leqslant 2 + w_{ijr}^2, \quad \forall r \in U; i \in J_r; j \in J_r\, i \neq j \tag{12}$$

$$\sum_{\substack{j \in J_r \\ j \neq i}} w_{ijr}^2 \leqslant u_r - 1, \quad \forall r \in U; i \in J_r \tag{13}$$

$$C_i^1 + L_i \leqslant C_j^1 + L_j + G^*(1 - z_{ij}^2), \quad \forall j \in J; i \in J\, i \neq j \tag{14}$$

$$z_{ij}^2 + z_{ji}^2 = 1, \quad \forall j \in J; i \in J\, i \neq j \tag{15}$$

$$E_{max} \geqslant (C_j^1 + L_j) - (C_i^1 - P_i^1 - S_i) - z_{ji}^{1*}G, \quad \forall j \in J; i \in J\, i \neq j : P_j^1 + S_j + L_j > E_{max} \tag{16}$$

$$z_{ij}^2 + z_{ij}^1 \leqslant z_{ij}^3 + 1, \quad \forall i \in J; j \in J\, j \neq i \tag{17}$$

$$\sum_{\substack{j \in J \\ j \neq i}} z_{ij}^3 \leqslant m^1 - 2, \quad \forall i \in J : P_i^1 + S_i + L_i > E_{max} \tag{18}$$

$$C_{max} \geqslant C_j^1 + L_j, \quad \forall j \in J \tag{19}$$

$$x_{jk}^h \in \{0,1\}, \quad \forall h = 1,2; j \in J; k \in K_j^h \tag{20}$$

$$y_{ij}^h \in \{0,1\}, \quad \forall h = 1,2; j \in J; i \in J\, i \neq j \tag{21}$$

$$w_{ijr}^1, w_{ijr}^2 \in \{0,1\}, \quad \forall r \in U; j \in J_r; i \in J_r\, i \neq j \tag{22}$$

$$z_{ij}^1, z_{ij}^2, z_{ij}^3 \in \{0,1\}, \quad \forall j \in J; i \in J\, i \neq j \tag{23}$$

$$C_i^h \geqslant 0, \quad \forall h = 1,2; i \in J \tag{24}$$

$$C_{max} \geqslant 0, \tag{25}$$

The objective function (1) minimizes the closing time of the last OR in use. Constraint (2) requires all surgeries being assigned to only one OR and one bed of PACU. Constraint (3) states that the surgery i precedes surgery j in stage h, if the completion time of i is lesser than or equal to the completion time of surgery j in stage h. Constraint (4) requires a single predecessor for each combination of two surgeries. Constraint (5) sets the completion time of any surgery to a value greater or equal than the sum of the setup time of the surgery plus the duration of the surgical act. Constraint (6) is the set of constraints that determines the precedence between surgeries that are assigned to a same OR. Constraint (7) determines the precedence between surgeries that are assigned to a same bed of PACU. Constraint (8) forces no waiting times to move from stage 1 to stage 2, whereas (9) enforces priorities between operations. Constraint (10) determines the order in which each surgeon performs her/his assigned surgeries. Constraint (11) forces the start of surgery that uses a determined resource to never start before the completion of another surgery using the same resource. Constraint (12) avoids surgeries using the same resource to be performed at the same time or in parallel, while (13) states that the maximum number of surgeries programmed in parallel is given by the number of available resources. Constraint (14) determines the precedence between the completions of the cleaning activities after the surgeries. For each combination of surgeries, constraint (15) states that there is only one predecessor. Constraint (16) requires the difference between the end of surgery j and the start of surgery i to be lesser than or equal to E_{max}. Constraint (17) determines whether the time of the surgeries scheduled in parallel is greater than E_{max}. Constraint (18) ensures that E_{max} is met. Constraint (19) indicates that C_{max} is greater or equal to the closing time of any ORs. Finally, Constraints (20)–(25) define the domain of the variables.

4.2. Constraint programming model

The CP model is formulated using the same indexes, parameters and sets described for the MILP. Additionally, it uses the following variables:

x_j^h = OR(bed) where is programmed the surgery j (h = 1 is OR, h = 2 is bed).
$C_{start\,j}^h$ = start time of surgery j in the stage h.
$C_{end\,j}^h$ = end time of surgery j in the stage h.

The CP uses the same objective function (1) as the MILP, and the following are the sets of constraints:

$$C_{start\,j}^1 + P_j^1 + S_j = C_{end\,j}^1, \quad \forall j \in J \tag{26}$$

$$C_{start\,j}^2 + P_j^2 = C_{end\,j}^2, \quad \forall j \in J \tag{27}$$

$$C_{start\,j}^2 = C_{end\,j}^1, \quad \forall j \in J \tag{28}$$

$$\text{if } x_j^1 = x_i^1 \text{ then } C_{end\,j}^1 + L_j \leqslant C_{start\,i}^1 \vee C_{end\,i}^1 + L_i \leqslant C_{start\,j}^1, \quad \forall j \in J; i \in J\, i \neq j \tag{29}$$

$$\text{if } x_j^2 = x_i^2 \text{ then } C_{end\,j}^2 \leqslant C_{start\,i}^2 \vee C_{end\,i}^2 \leqslant C_{start\,j}^2, \quad \forall j \in J; i \in J\, i \neq j \tag{30}$$

$$\text{if } x_j^1 = x_i^1 \text{ then } C_{end\,j}^1 + L_j \leqslant C_{start\,i}^1, \quad \forall j \in B; i \in B_j \tag{31}$$

$$C_{end\,j}^1 + S_j^a \leqslant C_{start\,i}^1 + S_i \vee C_{end\,i}^1 + S_i^a \leqslant C_{start\,j}^1 + S_j, \quad \forall a \in M; j \in M_a; i \in M_a\, i \neq j \tag{32}$$

$$u_r - 1 \geqslant \sum_{\substack{i \in J_r \\ i \neq j}} \Big(\text{if } C_{start\,j}^1 < C_{end\,i}^1 + S_r \wedge C_{start\,i}^1 < C_{end\,j}^1 + S_r \wedge C_{end\,j}^1 \leqslant C_{end\,i}^1 \text{ then } 1 \Big), \quad \forall r \in U; j \in J_r \tag{33}$$

$$m^1 - 2 \geqslant \sum_{\substack{i \in J \\ i \neq j}} \Big(\text{if } C_{end\,j}^1 + L_j - C_{start\,i}^1 > E_{max} \wedge C_{end\,j}^1 \leqslant C_{end\,i}^1 \text{ then } 1 \Big), \quad \forall j \in J : P_j^1 + T_j + L_j > E_{max} \tag{34}$$

$$C_{end\,j}^1 + L_j \leqslant C_{max}, \quad \forall j \in J \tag{35}$$

$$x_j^h \in K_j^h, \quad \forall j \in J; h = 1,2 \tag{36}$$

$$C_{start\,j}^h, C_{end\,j}^h \geqslant 0, \quad \forall h = 1,2; j \in J \tag{37}$$

The set of constraints in (26) states that the surgery end time is its start time plus the duration of the preparation and the surgical act duration. Constraint (27) indicates that the completion time of surgery in stage 2, is the start of the post-anesthesia recovery plus the recovery time. Constraint (28) is the no wait constraint between stages one and two, for any surgery. Constraint (29) indicates that, if two operations are scheduled in the same OR, one must precede the other. Constraint (30) is similar to (29), for the recovery phase. Constraint (31) sets priorities between surgeries. Constraint (32) indicates that a surgeon can only perform one surgery at a time. Constraint (33) indicates that the number of units of a resource used simultaneously depends on the available units for that resource. Constraint (34) restricts the maximum waiting time for emergency surgeries. Constraint (35) forces C_{max} to be equal to

the closing time of the latest ORs. Finally, Constraints (36), (37) declare the domain of the decision variables.

5. Metaheuristic

Due to the complexity of the problem (it is NP Hard), a metaheuristic is proposed, based on a genetic algorithm (GA) and a constructive heuristic (CH), which finds feasible solutions to large-scale instances within a reasonable execution time.

The main idea is as follows: the GA generates a *population* consisting of vectors representing possible sequences in which the surgeries could be assigned to ORs. For each sequence, the CH computes the corresponding values of the starting and ending times of all the surgeries (vector elements), assigns resources and post-anesthesia beds to them according to availability, checks for feasibility, and computes the sequence's objective value. Each GA-generated sequence, completed with the values computed by the CH, is a feasible solution to the problem. In order to improve the quality of the population, the GA replaces old solutions with new ones, if these have a better objective value. Once the stopping condition is met, the best solution is selected among all the solutions in the population.

5.1. Genetic algorithm

The GA generates an initial population by randomly performing permutations on a vector containing all the surgeries. Each permutation or GA generated solution represents a sequence in which surgeries can be assigned to ORs.

In each iteration, the GA randomly chooses two solutions from the current population and *crosses* or combines them to generate two new solutions. One of these new solutions is selected as a candidate to replace an old solution. With a low probability, the GA applies a *mutation* operator to this candidate solution, consisting of trying all exchange options between any two of its elements (surgeries) and evaluating the objective each time. The best resulting solution is stored and becomes the new candidate solution. The candidate solution replaces the worst solution in the current population if its objective value is better, maintaining a fixed size population.

We use two stopping conditions: a number of consecutive iterations after which there has been no improvement in the value of the objective function and a maximum of total iterations.

The crossing between two solutions in the reproductive stage of the GA is the linear order crossover described in Falkenauer and Bouffouix (1991). Let vectors $vect_1$ and $vect_2$ be the solutions selected for the crossover and $vect_4$ and $vect_5$ the resulting solutions after the crossover. Two cut points within vectors $vect_1$ and $vect_2$ are selected randomly, $cut1$ and $cut2$. All elements of $vect_1$ between these cuts are preserved in $vect_4$, in the same positions. The remaining elements of $vect_4$ are filled using elements of $vect_2$, as follows: pick the element of $vect_2$ to the right of $cut2$ and replace the same element of $vect_4$, if the surgery represented by that element has not been already included in $vect_4$. Iterate taking each time the next element to the right in both vectors, and when the end of the vector is reached, do the same starting from the first element until $cut1$ is reached. In the example of Fig. 2, the elements between $cut1$ and $cut2$, third and fourth elements of $vect_1$, are preserved in $vect_4$. Then, starting from $cut2$, the value e from $vect_2$ should replace the same position of $vect_4$, but since e already belongs to $vect_4$, the procedure moves to the next element to the right in the origin vector $vect_2$. In this case, there are no more elements to the right in $vect_2$, and the procedure jumps to the first element of $vect_2$, which is a. Since surgery a is already contained in $vect_4$, the next element in $vect_2$, b, is used to fill the position in $vect_4$. The following elements of $vect_4$ are filled with the next elements of $vect_2$, c and d.

5.2. Constructive heuristic

The CH performs all the calculations required to compute the value of the objective function for each sequence of surgeries generated by the GA. The CH consists of the following 7 steps. The first 6 steps are applied to each surgery or element of the sequence, while the seventh step is applied once the 6 first steps have been applied to all surgeries in the sequence:

Step 1: Starting Time of Surgery. Assign the surgery to the least used OR. Set the starting time of the surgery.
Step 2: Availability of surgeon. Delay (if required) the starting time of the surgery just assigned, to meet the soonest time at which the surgeon is available.
Step 3: Availability of Resources. Delay (if required) the starting time of the surgery to meet the soonest time at which all the resources are available.
Step 4: Maximum waiting time. Delay (if required) the starting time of the surgery to meet E_{max}.
Step 5: Availability of Bed in the PACU. Delay (if required) the starting time of the surgery to meet the soonest time at which a post-anesthesia bed is available.
Step 6: Time Allocation. Compute the occupancy time for all resources (OR, bed, nurse, etc.).
Step 7: Compute C_{max}. If there are violated priorities between surgeries, repair the scheduling and recalculate the times. Otherwise, compute C_{max}.

5.3. Complete metaheuristic

Fig. 3 presents the pseudo-code of the complete metaheuristic, including GA and CH.

6. Computational experiments

The master group of surgeries shown in Table 2 was generated randomly. The master group includes information on the clinical specialty to which each surgery belongs, the resources required at the time of the operation and the assigned surgeon.

The duration of the surgeries in master Table 2, shown in Table 3, was generated randomly using a log-normal distribution, along with the information presented in Table 3, taken from Marcon, Kharraja, Smolski, Luquet, and Viale (2003). To generate setup times and cleaning times, the methodology of Jebali et al. (2006) was used. If the duration of surgery was less than 90 min, the preparation time of the OR was set to 10 min, otherwise to 20 min. The preparation and anesthesia time of the patient was 10 min if the surgical act duration was less than 60 min, 20 min if the surgical act lasted between 60 and 120 min, and 30 min otherwise. If the duration of surgical act was less than 20 min, the OR cleaning lasted 15 min, 30 min otherwise. The recovery time post-anesthesia is generated using a log-normal distribution with a mean equal to the duration of surgery minus 10 min, and a standard deviation of 15 min (Jebali et al., 2006).

Furthermore, if the time of surgical act was less than 60 min, a time of preparation of the surgeon was set to 15 min. If the time of surgical act was between 60 and 120 min, the preparation time of the surgeon was 30 min, and 45 min otherwise. Preparation time of resources was generated randomly between 15 and 90 min.

All times were rounded to multiples of 5. For example, if the random time value obtained is 33.8 min, it was rounded to 35 min. This is consistent with what is done in practice. The

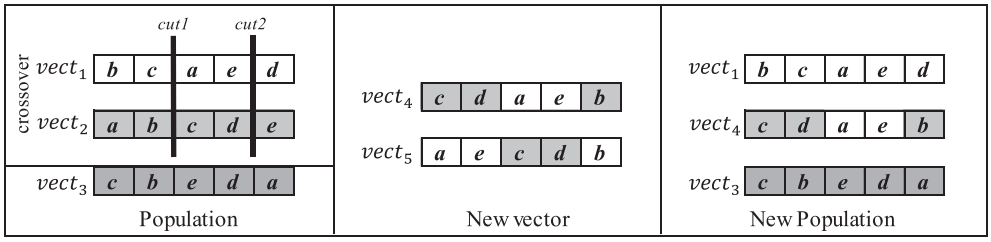

Fig. 2. Example GA.

Generate an initial population (set of solution vectors). Evaluate the solutions using **CH function**.

While (the stopping condition is not met) Do
- **Select** randomly two solutions for crossover.
- Apply the **crossing operator** to the solutions, obtaining two new solutions.
- Save one of the new solutions as **solution candidate**. Evaluate using the **CH function**.
- With a low probability, apply the **mutation operator** to all pairs of elements in the **solution candidate**. Evaluate using the **CH function**. Save the best as **solution candidate**
- Replace the worst solution in the population with the **solution candidate** if its value is better

Supply and save the **best solution** obtained.

CH function:
For each surgery in the sequence
- Apply step 1 to step 6.

Apply step 7.
Return the solution to GA.

Fig. 3. Heuristic Pseudo-code.

Table 2
Master set of surgeries to be scheduled.

Type of surgery	Id. Surgery	Resources			Surgeon	Type of surgery	Id. Surgery	Resources			Surgeon
General	1	1	2		1	Ambulatory	21				17
	2				2		22	1	10	11	13
	3	2	3		3		23	8	7		14
	4	1			4		24				15
	5	4			5	Orthopedics	25	13			18
	6				6		26	14	15		19
	7	3	5	6	7		27	14	15		20
	8				1		28				18
Endoscopy and radiology	9	1			8		29	6			19
	10	7	8		9	Otorhino-laryngology	30	4	5		21
	11				10		31	16			22
	12				11		32	16	17	18	23
	13	8	9	10	12		33				21
	14	10			8		34	17	18		22
	15				9		35	16			23
	16	9			10	Ophthalmology	36	4	18	19	25
Ambulatory	17	2	11		13		37				26
	18				14		38				25
	19	11	7		15		39	19			26
	20	12	13		16		40	19			27

number of ORs used was generated as the sum of the time of preparation, cleaning and surgical act divided by 600 min (10 h), i.e. $m^1 = (1/600) * \sum_{j \in J}(L_j + T_j + P_j^1)$. The number of post-anesthesia beds was generated randomly, varying in the range $m^1 - 1 \leqslant m^2 \leqslant 2m^1$. The number of resource units was generated randomly varying in the range $1 \leqslant u_r \leqslant |J_r|$.

30 instances were generated using the surgeries in the master set. These instances consider 15, 16, 17, 19, 21, 24, 27, 30, 35 and 40 surgeries, and for each number of surgeries, three instances were developed. For example, for instances 1, 2 and 3, 15 surgeries were selected randomly from Table 2, and so on. For instance 28, 29 and 30, all 40 surgeries were used.

Both MILP and CP models, including the heuristic, were programmed in C++, using Win32 Console Application of Visual Studio 2010. MILP and CP were solved with the CPLEX 12.6 package. All the algorithms were programmed with a run limit. The MILP was run with a time limit of 1 h (3600 s) and a limit of 8 h (28,800 s) for each instance. The CP was run with a limit of 1 h and 1,000,000 failures allowed before terminating a search. The heuristic was programmed to stop after 1500 iterations without changes

Table 3
Duration of surgeries.

Type of surgery	Mean duration of surgery	Standard deviation	Range
General	180	60	[30–420]
Endoscopy and radiology	30	10	[20–120]
Ambulatory	30	10	[10–60]
Orthopedics	120	30	[45–200]
Otorhinolaryngology	45	30	[20–180]
Ophthalmology	60	15	[30–120]

in the population or a maximum of 10,000 total iterations. The computational tests were performed on a computer with operating system Windows 7 Professional 64-bit, a processor Intel (R) Core (TM) i7-2600 3.40 GHz CPU @ 3.40 GHz and 16 GB RAM. All tests were performed using only one processor core.

Table 4 shows some of the results of the computational tests using the MILP, CP and heuristic. The first column of Table 4 shows the number of the instance while the second shows the total number of scheduled surgeries considered in that instance. The objective value, run time and gap value obtained with the MILP, for a time limit of 1 h, is shown in the third, fourth and fifth column. The sixth column shows the gap between the solution of the MILP with the one-hour limit, and the best gap available (obtained by using MILP with a time limit of 8 h). The seventh eighth and ninth columns show the objective value, run time and gap obtained with the MILP when the time limit was of 8 h. The objective value, gap (between MILP bounds and CP solutions) and the run time of CP are shown in the tenth, eleventh and twelfth column, while the thirteenth column shows the type of solution obtained (feasible or optimal). The fourteenth, fifteenth and sixteenth columns show the objective, gap and CP run time that were executed with a limit value of 1 h. The values of the objective, gap (between MILP bounds and heuristic solutions) and run times for the heuristic are shown in the remaining columns. The MILP reached optimal values before the time limit in 19 of the 30 instances. By comparing instances that were stopped at the 1 and 8 h limits, an average change of 1.7% is observed in the gap, with a range of variation in the value of the objective of 0–35 min. In general, seven additional hours of running do not improve significantly the value of the objective, except for instances 8, 14 and 21.

CP shows optimality in 10 instances before the limits are reached. However, it achieves the same objective value as the MILP whenever the MILP reaches optimality. However, in instance 19, the objective value for both MILP and CP is 830, but only MILP can prove optimality. When using a limit of 1,000,000 failures, an average solution time of 1265 s was required, whereas with a one hour time limit, an average resolution time of 2400 s was obtained. If both limits are combined, an average run time of 1000 s is obtained. In all these cases, the solutions are very similar. In only five of the instances, the objective values are different, and even when both types of limits are imposed, the objective values (C_{max}) never increase by more than 10 min as compared to imposing only time or only failure limits. Comparing the CP with the MILP (both executed for 1 h), MILP gets better solutions than CP in 6 instances, while CP is better in 3 instances. The largest difference occurs in instance 8, when the CP overcomes the MILP.

The heuristic obtained very good solutions in a very short run time. It found the optimal solution in 18 of the 19 solutions shown to be optimal by the MILP. The heuristic even achieved better solutions than MILP with a time limit of 8 h in instances 13 and 21.

Table 4
Comparison of solution methods.

Instance	Number of surgeries	MILP (3600 s)				MILP (28,800 s)			CP (Fail Limit)				CP (3600 s)			Heuristic		
		Obj. (min)	Time (s)	Gap (%)	Gap ** (%)	Obj. (min)	Time (s)	Gap (%)	Obj. (min)	Time (s)	Gap ** (%)	Sol.	Obj. (min)	Time (s)	Gap ** (%)	Obj. (min)	Time (s)	Gap ** (%)
1	15	570	15.3	0.0	*	*	*	*	*	414.6	*	Feas	*	Limit	*	*	1.0	*
2	15	505	Limit	4.0	*	*	Limit	*	510	454.9	4.9	Feas	*	*	*	*	1.7	*
3	15	570	Limit	26.3	23.5	*	Limit	23.5	590	393.5	26.1	Feas	585	*	25.5	590	1.4	26.1
4	16	545	Limit	37.1	30.7	*	Limit	30.7	560	599.9	32.6	Feas	560	*	32.6	550	1.6	31.3
5	16	515	Limit	17.5	*	*	Limit	*	525	623.0	19.1	Feas	525	*	19.1	520	1.5	18.3
6	16	525	Limit	17.1	*	*	Limit	*	530	444.0	17.9	Feas	530	*	17.9	*	2.1	*
7	17	560	Limit	1.0	*	*	Limit	*	575	680.0	3.6	Feas	575	*	3.6	565	3.3	1.9
8	17	585	Limit	26.5	*	550	Limit	21.8	561	583.0	23.3	Feas	560	*	23.2	555	4.2	22.5
9	17	625	9.6	0.0	*	*	*	*	*	4.1	*	Opt	*	4.1	*	*	1.4	*
10	19	685	14.4	0.0	*	*	*	*	*	1.6	*	Opt	*	1.6	*	*	1.3	*
11	19	670	6.2	0.0	*	*	*	*	*	2.5	*	Opt	*	2.5	*	*	1.5	*
12	19	565	Limit	13.4	*	*	Limit	*	575	406.3	14.9	Feas	575	*	14.9	*	2.7	*
13	21	605	Limit	16.5	*	*	Limit	*	615	945.0	17.9	Feas	*	*	*	595	4.0	15.1
14	21	665	Limit	22.6	*	645	Limit	20.2	655	1060.5	21.4	Feas	655	*	21.4	650	5.0	20.8
15	21	805	3.6	0.0	*	*	*	*	*	0.9	*	Opt	*	0.9	*	*	1.7	*
16	24	770	54.5	0.0	*	*	*	*	*	1.6	*	Opt	*	1.6	*	*	2.5	*
17	24	560	120.1	0.0	*	*	*	*	*	8.1	*	Opt	*	8.1	*	*	3.4	*
18	24	600	58.8	0.0	*	*	*	*	*	6.4	*	Opt	*	6.4	*	*	6.6	*
19	27	830	77.0	0.0	*	*	*	*	*	1790.6	*	Feas	*	Limit	*	*	5.1	*
20	27	705	507.5	0.0	*	*	*	*	*	2566.4	*	Feas	*	Limit	*	720	6.4	2.1
21	27	575	Limit	40.9	40.8	555	Limit	38.7	571	1704.7	40.4	Feas	570	*	40.3	545	8.4	37.6
22	30	705	143.2	0.0	*	*	*	*	*	2801.2	*	Feas	*	*	*	*	14.8	*
23	30	1165	311.6	0.0	*	*	*	*	*	5562.3	*	Feas	*	*	*	*	5.0	*
24	30	975	39.8	0.0	*	*	*	*	*	2078.4	*	Feas	*	*	*	*	5.7	*
25	35	765	320.2	0.0	*	*	*	*	*	22.6	*	Opt	*	22.6	*	*	18.5	*
26	35	865	145.9	0.0	*	*	*	*	*	15.5	*	Opt	*	15.5	*	*	16.8	*
27	35	945	102.6	0.0	*	*	*	*	*	3450.5	*	Feas	*	*	*	*	9.7	*
28	40	815	1204.4	0.0	*	*	*	*	*	5945.3	*	Feas	*	*	*	*	31.0	*
29	40	780	472.9	0.0	*	*	*	*	*	21.5	*	Opt	*	21.5	*	*	23.2	*
30	40	730	1072.0	0.0	*	*	*	*	*	5378.4	*	Feas	*	*	*	*	30.2	*

* Identical values to those obtained by the MILP with a time limit of 3600 s.
** Gap calculated using the MILP bounds with a time limit of 28,000 s.

Table 5
Robustness of the solution.

Instance	Number of surgeries	Surgery delay		Extra waiting time		Delay in C_{max}	
		Mean	S.D.	Mean	S.D.	Mean	S.D.
3	15	13.1	30.3	4.5	16.0	67.0	51.3
9	17	19.4	41.7	1.5	3.0	49.5	58.3
17	24	23.4	46.5	0.9	2.2	65.5	48.6
22	30	23.1	52.7	0.0	0.5	58.2	53.2
30	40	28.9	53.4	0.0	0.5	63.6	57.8

For comparison purposes, the problem was solved in three relaxed versions: without consideration of the emergency surgeries, i.e., without caring for having intervals of at most one hour; without the second stage (recovery), and without resources. By not considering emergency surgeries, 47% of the solutions violate the E_{max}. When it did not consider the recovery beds, 74% of the instances had a schedule conflict with respect to the availability of beds. By not considering the resources, 100% of the solutions had conflicts regarding resource demand. These relaxed solutions show the relevance of considering all these factors together.

As the programming in our method was obtained using average surgery times, we analyzed the robustness of the solution considering the uncertainty in the duration of the surgeries. We generated 2000 realizations of the duration of the surgeries, using a Monte Carlo method. The arrival times of up to 4 emergency surgeries in a day were assumed uniformly distributed between 0 and C_{max}. The results were analyzed, keeping fixed the scheduling obtained by the MILP, CP model or heuristics. The analysis focused on the delay of each surgery with respect to its scheduled starting time, the waiting time that an emergency surgery can suffer in excess of E_{max}, and the delay in the closing time of the last OR (C_{max}). Table 5 shows the mean and standard deviation of the results for five instances of different sizes.

As Table 5 shows, when considering uncertainty in surgery times the average closing time is longer than the value obtained deterministically in up to approximately one hour, with a standard deviation that is also close to one hour. As the number of scheduled surgeries increases, the delay in relation to the scheduled start time of each surgery tends to increase. The waiting time of emergency surgeries barely exceeds the maximum time E_{max}. This is because the programming distributes short surgeries among the different ORs and in time, avoiding blocking simultaneously all ORs for too long times. Moreover, as the size of the instance grows, so does the number of ORs, increasing the probability of having an OR available on arrival of an emergency surgery.

7. Conclusions

Efficient scheduling of operating rooms is extremely difficult, and any improvement in the process could mean significant savings. Usually, this problem is solved sequentially, i.e., surgeries are assigned to ORs in a first stage. In a second stage, the equipment and other resources are assigned to the surgeries and, finally, the recovery beds are added to the programming. Solving this problem sequentially does not allow finding the bottlenecks in the programming until the last stage is solved and the recovery beds assigned. In this paper we develop several tools to improve the scheduling, that address the issue more comprehensively, while addressing all stages at once, considering not only the assignment of surgeries to operating rooms, but all the resources required for each surgery (human and material), and the recovery beds. Jointly addressing all these issues not only reduces bottlenecks, but allows finding a truly optimal solution to the problem. Furthermore, we include in the programming the emergency surgeries, through a deterministic solution that considers spacing the planned inter-surgery times by no more than one hour. This procedure assures prompt attention to any emergency surgery that might be required. The developed tools include a MILP model, a CP model and an ad-hoc heuristic, whose results are compared in run time and quality. Finally, we perform a probabilistic analysis of the problem, by generating a number of realizations and checking the performance of the procedure.

The CP model delivers good quality solutions in run times that are, in general, shorter than those of the MILP. However, the improvement is not significant. The metaheuristic, on the other hand, delivers solutions of very good quality in short run times, with averages of 7 s, on instances ranging from 15 to 40 surgeries requiring scheduling. The gap between MILP solutions and heuristic solutions never exceeded 3.6%, while in some instances, the solutions found by the heuristic were better than solutions obtained by using time-limited MILP in up to a 5.2%. With these short run times and good results, this heuristic could also be used for online surgery rescheduling, with small modifications.

A natural next step is explicitly including uncertainty in the duration of the activities and arrival of emergency surgeries.

Acknowledgements

We thank the valuable comments made by a referee, which helped significantly improving the quality of the paper. Lüer-Villagra and Marianov gratefully acknowledge support by the Institute Complex Engineering Systems through grants ICM P-05-004-F and CONICYT FBO16.

References

Addis, B., Carello, G., Grosso, A., & Tànfani, E. (2015). Operating room scheduling and rescheduling: A rolling horizon approach. *Flexible Services and Manufacturing Journal*, 1–27.

Augusto, V., Xie, X., & Perdomo, V. (2010). Operating theatre scheduling with patient recovery in both operating rooms and recovery beds. *Computers & Industrial Engineering, 58*(2), 231–238.

Cardoen, B., Demeulemeester, E., & Beliën, J. (2009a). Optimizing a multiple objective surgical case sequencing problem. *International Journal of Production Economics, 119*(2), 354–366.

Cardoen, B., Demeulemeester, E., & Beliën, J. (2009b). 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., & Beliën, J. (2010). Operating room planning and scheduling: A literature review. *European Journal of Operational Research, 201*(3), 921–932.

Dekhici, L., & Belkadi, K. (2010). Operating theatre scheduling under constraints. *Journal of Applied Sciences, 10*(14), 1380–1388.

Dios, M., Molina-Pariente, J. M., Fernandez-Viagas, V., Andrade-Pineda, J. L., & Framinan, J. M. (2015). A decision support system for operating room scheduling. *Computers & Industrial Engineering, 88*, 430–443.

Erdem, E., Qu, X., & Shi, J. (2012). Rescheduling of elective patients upon the arrival of emergency patients. *Decision Support Systems, 54*(1), 551–563.

Falkenauer, E., & Bouffouix, S. (1991). A genetic algorithm for job shop. In *IEEE international conference on robotics and automation* (pp. 824–829).

Fei, H., Chu, C., & Meskens, N. (2009). 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.

Fei, H., Meskens, N., & Chu, C. (2010). A planning and scheduling problem for an operating theatre using an open scheduling strategy. *Computers & Industrial Engineering, 58*(2), 221–230.

Fleet, R., & Poitras, J. (2011). Have we killed the golden hour of trauma? *Annals of Emergency Medicine, 57*(1), 73–74.

Guerriero, F., & Guido, R. (2011). Operational research in the management of the operating theatre: A survey. *Health Care Management Science, 14*(1), 89–114.

Guinet, A., & Chaabane, S. (2003). Operating theatre planning. *International Journal of Production Economics, 85*(1), 69–81.

Guo, M., Wu, S., Li, B., & Rong, Y. (2014). Maximizing the efficiency of use of nurses under uncertain surgery durations: A case study. *Computers & Industrial Engineering, 78*, 313–319.

Gupta, J. N. D. (1988). Two-stage, hybrid flowshop scheduling problem. *Journal of the Operational Research Society, 39*(4), 359–364.

Jebali, A., Hadj Alouane, A. B., & Ladet, P. (2006). Operating rooms scheduling. *International Journal of Production Economics, 99*(1–2), 52–62.

Lee, S., & Yih, Y. (2014). Reducing patient-flow delays in surgical suites through determining start-times of surgical cases. *European Journal of Operational Research, 238*(2), 620–629.

Marcon, E., & Dexter, F. (2006). Impact of surgical sequencing on post anesthesia care unit staffing. *Health Care Management Science, 9*(1), 87–98.

Marcon, E., Kharraja, S. d., Smolski, N., Luquet, B., & Viale, J. P. (2003). Determining the number of beds in the postanesthesia care unit: A computer simulation flow approach. *Anesthesia & Analgesia, 96*(5), 1415–1423.

Marques, I., Captivo, M., & Vaz Pato, M. (2012). An integer programming approach to elective surgery scheduling. *OR Spectrum, 34*(2), 407–427.

Marques, I., Captivo, M. E., & Vaz Pato, M. (2014). Scheduling elective surgeries in a Portuguese hospital using a genetic heuristic. *Operations Research for Health Care, 3*(2), 59–72.

May, J. H., Spangler, W. E., Strum, D. P., & Vargas, L. G. (2011). The surgical scheduling problem: Current research and future opportunities. *Production and Operations Management, 20*(3), 392–405.

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

Newgard, C. D., Schmicker, R. H., Hedges, J. R., Trickett, J. P., Davis, D. P., Bulger, E. M., ... Nichol, G. (2010). Emergency medical services intervals and survival in trauma: Assessment of the "golden hour" in a North American prospective cohort. *Annals of Emergency Medicine, 55*(3), 235–246.

Paoletti, X., & Marty, J. (2007). Consequences of running more operating theatres than anaesthetists to staff them: A stochastic simulation study. *British Journal of Anaesthesia, 98*(4), 462–469.

Perdomo, V., Augusto, V., & Xiaolan, X. (2006). Operating theatre scheduling using lagrangian relaxation. *International conference on service systems and service management* (Vol. 2, pp. 1234–1239). .

Persson, M., & Persson, J. A. (2009). Health economic modeling to support surgery management at a Swedish hospital. *Omega, 37*(4), 853–863.

Pham, D. N., & Klinkert, A. (2008). Surgical case scheduling as a generalized job shop scheduling problem. *European Journal of Operational Research, 185*(3), 1011–1025.

Ribas, I., Leisten, R., & Framinan, J. M. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research, 37*(8), 1439–1454.

Roland, B., di Martinelly, C., & Riane, F. (2006). Operating theatre optimization: A resource-constrained based solving approach. *International conference on service systems and service management* (Vol. 1, pp. 443–448). .

Roland, B., Di Martinelly, C., Riane, F., & Pochet, Y. (2010). Scheduling an operating theatre under human resource constraints. *Computers & Industrial Engineering, 58*(2), 212–220.

Ruiz, R., & Vazquez-Rodriguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research, 205*(1), 1–18.

van Essen, J. T., Hans, E. W., Hurink, J. L., & Oversberg, A. (2012). Minimizing the waiting time for emergency surgery. *Operations Research for Health Care, 1*(2–3), 34–44.

van Essen, J. T., Hurink, J. L., Hartholt, W., & van den Akker, B. J. (2012). Decision support system for the operating room rescheduling problem. *Health Care Management Science, 15*(4), 355–372.

Vijayakumar, B., Parikh, P. J., Scott, R., Barnes, A., & Gallimore, J. (2013). A dual bin-packing approach to scheduling surgical cases at a publicly-funded hospital. *European Journal of Operational Research, 224*(3), 583–591.

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