A two level Metaheuristic for the Operating Room Scheduling and Assignment Problem

Published version:

DOI:10.1016/j.cor.2014.08.014

Author's Accepted Manuscript

www.elsevier.com/locate/caor

A two level metaheuristic for the operating room scheduling and assignment problem

Roberto Aringhieri, Paolo Landa, Patrick Soriano, Elena Tànfani, Angela Testi

PII: S0305-0548(14)00224-X
DOI: http://dx.doi.org/10.1016/j.cor.2014.08.014
Reference: CAOR3640

To appear in: *Computers & Operations Research*

Cite this article as: Roberto Aringhieri, Paolo Landa, Patrick Soriano, Elena Tànfani, Angela Testi, A two level metaheuristic for the operating room scheduling and assignment problem, *Computers & Operations Research,* http://dx.doi.org/10.1016/j.cor.2014.08.014

A two level Metaheuristic for the Operating Room Scheduling and Assignment Problem

Roberto Aringhieri[a,*], Paolo Landa[b], Patrick Soriano[c], Elena Tànfani[b], Angela Testi[b]

[a] *Department of Computer Science, University of Torino, Italy*
[b] *Department of Economics and Business Studies, University of Genova, Italy*
[c] *Department of Management Sciences and CIRRELT, HEC Montréal, Canada*

Abstract

Given a surgery department comprising several specialties that share a fixed number of operating rooms and post-surgery beds, we study the joint operating room (OR) planning and advanced scheduling problem. More specifically, we consider the problem of determining, over a one week planning horizon, the allocation of OR time blocks to specialties together with the subsets of patients to be scheduled within each time block. The aim of this paper is to extend and generalize existing approaches for the joint OR planning and scheduling problem. First, by allowing schedules that include patients requiring weekend stay beds which was not the case previously. Second, by tackling simultaneously both the OR planning and patient scheduling decision levels, instead of taking them into account in successive phases. To achieve this, we exploit the inherent hierarchy between the two decision levels, i.e., the fact that the assignment decisions of OR time blocks to surgical specialties directly affect those regarding the scheduling of patients, but not the reverse. The objective function used in this study is an extension of an existing one. It seeks to optimize both patient utility (by reducing waiting time costs) and hospital utility (by reducing production costs measured in terms of the number of weekend stay beds required by the surgery planning).

*Corresponding author

Email addresses: roberto.aringhieri@unito.it (Roberto Aringhieri), paolo.landa@unige.it (Paolo Landa), patrick.soriano@hec.ca (Patrick Soriano), etanfani@economia.unige.it (Elena Tànfani), testi@economia.unige.it (Angela Testi)

$0-1$ linear programming formulations exploiting the stated hierarchy are proposed and used to derive a formal proof that the problem is NP-hard. A two level metaheuristic is then developed for solving the problem and its effectiveness is demonstrated through extensive numerical experiments carried out on a large set of instances based on real data.

Keywords: Operating Room planning, advanced scheduling, $0-1$ model, Metaheuristic
2000 MSC: 90C11, 90C27, 90C59

---

1. Introduction and literature review

Operating Rooms (ORs) planning is a critical activity with important financial impacts for most hospital setting. In addition, demand for surgery very often overwhelms supply therefore causing long waiting times for patients and reducing their quality of life [1]. This is particularly true in publicly funded health care systems such as those found in Italy, in the province of Québec in Canada and many other settings. One of the main questions health care system planners and administrators are faced with when planning ORs is how can demand and supply meet, i.e., how should the available OR capacity be allocated in order to improve efficiency and productivity and how can efficiency be attained and measured. The review of the operations research and management science scientific literature clearly reveals an increasing interest of researchers towards OR planning and scheduling problems [2, 3].

Researchers frequently distinguish between strategic (long term), tactical (medium term) and operational (short term) decisions in order to better characterize their planning or scheduling problem even if there are no clear and universally accepted definitions of these three decision levels [2]. In the following, we concentrate our analysis on the OR planning and scheduling problem at both the tactical and operational levels under the block scheduling or closed block planning paradigm. When planning with this paradigm, each specialty is assigned a number of OR time blocks (usually with homogeneous block lengths of half-day or full day) for each planning period, typically a week to two weeks. Each specialty then schedules their surgical cases within these time blocks [4].

The OR planning and scheduling problem under the block scheduling approach can be viewed as being made up of three phases corresponding to three decision levels [5]. In the first phase, the problem addressed is that of determining, at a strategic level, the number and type of ORs available, the hours of operation of the ORs and how overall OR capacity is to be divided among surgical specialties, individual surgeons or groups [6, 7, 8, 9, 10]. Then, a cyclic timetable, often referred to as the "Master Surgical Schedule" (MSS), is constructed on a medium term horizon to define the specific assignment of OR blocks to specialties. The MSS must of course be updated whenever the total amount of OR time changes or when the make up of some specialties change. This can occur not only as a response to long term changes in the overall OR capacity or fluctuations in staffing, but also in response to seasonal fluctuations in demand [11, 12, 13, 14, 15, 16, 17]. The last phase, which may be called "surgery process scheduling", is generally separated into two sub-problems referred to as "advance scheduling" and "allocation scheduling" [18]. The first sub-problem consists in assigning a specific surgery and OR time block to each patient over the planning horizon, which can range from one week to one month [19, 20, 21, 22, 23, 24, 25]. Given this advanced schedule, the second sub-problem then determines the precise sequence of surgical procedures and the allocation of resources for each OR time block and day combination [26, 27, 28, 29, 30, 31] in order to implement it as efficiently as possible.

As evidenced by the references listed here-above, the vast majority of papers found in the literature only consider one decision level at a time. Approaches dealing with more than one planning level simultaneously are quite rare. Among these, Jebali et al. [32] use a two-phase approach to deal with both the advance scheduling and allocation scheduling problems and propose a 0-1 linear programming model aimed at minimizing OR overtime and under-time costs as well as hospitalization costs related to the number of days patients are kept in the hospital waiting for an operation or procedure. Testi et al. [5] present a hierarchical three-phase approach to determine operating theater schedules. First, integer programming models are developed in order to divide the available OR time among the different surgical specialties. Then they formulate a master surgery scheduling problem in order to assign a specific operating room and day of the planning horizon to the OR time blocks of each specialty. Finally, a discrete-event simulation model is used to evaluate the decisions concerning patients date, OR and time assignments. Tànfani and Testi [33] propose a 0-1 linear programming model to simultaneously address the decisions involved in the three phases of the OR planning and scheduling problem described above, excluding only the most strategic ones dealing with the number and type of the ORs and their operating hours. The objective of the model consists in minimizing a societal cost function that combines the patients' waiting time since referral and urgency status. The solution approach is based on a sequential heuristic. First, a subset of suitable patients is selected using heuristic rules. Then OR time blocks to which the selected subset of patients could be assigned considering their expected length of stay (LOS) are identified (i.e., time blocks in the schedule such that the patient would not require to stay hospitalized during the weekend). Finally, a reduced version of the $0-1$ linear programming model is solved. In that version, the patients and the OR blocks not selected in the previous two steps are excluded from consideration. The main limitation of this approach is that the decisions taken in the first two steps are not re-evaluated and therefore no interaction between them is considered nor any tradeoff investigated. In Choi and Wilhelm [34], they include in the analysis the problem of determining the the duration of time blocks reserved to each surgery sub-specialty and their sequencing, referred as Block Surgical Schedule (BSS). A newsvendor-based model has been developed to solve the BSS with the aim of minimizing the total expected lateness and earliness costs. Agnetis et al. [35] proposed a decomposition approach to solve MSS and assigning patients to available OR blocks. The solution of the two problems being done in sequence. The performance of the approach has been evaluated on a large set of real based instances and the solutions compared with those obtained by an exact integrated approach.

In this paper, we deal with the joint master surgical schedule and advanced scheduling problem but with the aim of extending and generalizing existing approaches as well as proposing more efficient solution methodologies. We assume that all strategic level decisions are given as input data, including the number of OR time blocks assigned to each specialty weekly. This responds to a real practical issue faced by surgery departments since these strategic decisions are generally the result of a long and complex negotiation process involving the different surgical specialties and the hospital administration. They are therefore not easily changed on a short term horizon. The first generalization consists in developing a modeling and solution approach which simultaneously considers both the OR planning and patient scheduling decision levels instead of taking them into account separately. This increases significantly the potential quality of the resulting schedules but, of course, at the expense of a significant increase in the difficulty for solving the problem. To achieve this generalization, we exploit the inherent hierarchy between the two decision levels, i.e., the fact that the assignment decisions of OR time blocks to surgical specialties directly affect those regarding the scheduling of patients, but not the reverse. The type of schedules considered here is also more general with respect to the tactical and operational level decisions than the ones produced in Tànfani and Testi [33] because it allows schedules in which patients may require weekend stay beds. Here, weekend stay beds are modeled as an additional limited resource that can be used but not exceeded. Finally, we adopt the idea proposed in [33] of using societal costs as the objective function but extending it to incorporate both patient utility (by reducing waiting time costs) and hospital utility (by reducing production costs measured in terms of the number of weekend stay beds required by the surgery planning). $0-1$ linear programming formulations exploiting the stated hierarchy are proposed and used to derive a formal proof that the problem is NP-hard.

As reported in [36], health care optimization problems are challenging, often requiring the adoption of unconventional solution methodologies. The solution approach proposed herein belongs to this family. It is a tabu search algorithm (see, e.g., [37, 38]) in which the main idea is to iterate the search between the two decision levels in such a way as to globally improve the solution. Its effectiveness is demonstrated through extensive numerical experiments carried out on a large set of instances based on real data.

The paper is organized as follows. Section 2 describes in more details the problem under investigation. Section 3 introduces $0-1$ formulations and presents a formal proof of complexity. The proposed solution algorithm is described in Section 4. Computational results highlighting the effectiveness of the two level approach as well as comparisons between the solutions produced by the metaheuristic and optimal ones are reported in Section 5. Concluding remarks and further research directions close the paper.

2. Problem definition and notation

Given a set of surgical specialties, a list of patients waiting to be operated on for each specialty and a number of OR time blocks to be assigned to each specialty, we face the problem of determining for a given planning horizon of one week: 1) the cyclic timetable that gives for each day of the planning horizon the assignment of specific OR time blocks to specialties, referred to as the *Master Surgical Schedule Problem* (MSSP); and 2) the surgery date and operating room assigned to each patient selected to be operated on, referred as *Surgical Case Assignment Problem* (SCAP). In the following, we will refer to this joint problem as the *Operating Room Planning Problem* (ORPP). Note that we assume as input data the strategic decisions pertaining to the number and length of OR time blocks available for surgery each day as well as the number of blocks assigned to each specialty. In accordance with the block scheduling paradigm, only a single specialty may be assigned to a given OR time block, i.e., OR blocks cannot be split among different specialties.

The subset of patients to be operated on among the waiting list and their order of admission is based on the prioritization system already introduced and validated in [39, 40] which is based on both the waiting time of the patient since its referral and its urgency status. A similar approach is also used in [41]. The objective pursued here is to minimize the total societal cost which includes both patient and hospital costs. Patient costs which depend on delays in meeting their clinical needs are explicitly included in the objective function which seeks to minimize the total priority score. Hospital costs, on the other hand, are treated indirectly by introducing capacity constraints controlling the number of beds that can be used over the weekend days (i.e., similar to a budget constraint), as will be described in more detail in Section 3.

The planning decisions have to satisfy many resource constraints related to OR time blocks (session) length, number of OR time blocks assigned to each surgical specialty, number of surgical teams available for each specialty and day, number of weekend stay beds available.

We assume that, as is the case in many publicly funded health systems, the number of patients on the waiting lists is greater than the maximum number of patients that can be operated on during the planning horizon considered. This means that we are concerned with the problem of selecting a subset of patients to be operated on among all the available patients. Note that in the specific setting of the partner hospital considered here, which is the same as in [33], emergency patients and outpatients (day surgery) are not considered as part of the elective patients to be scheduled since they use extra ORs dedicated for that purpose. Moreover, hospitalization beds on normal working days of the week (i.e., Monday to Friday) are considered as unlimited. Finally, like the majority of papers in this field (as discussed in [2]) we do not consider in the present analysis the uncertainty associated to effective surgery times and lengths of stay.

Notation. Let us introduce some necessary notation. Let I, J, and K be respectively the sets of patients, surgical specialties, and operating rooms, each indexed by the corresponding lower cased letter, i, j, and k. For simplicity, we will assume in the following that the surgery days within the week to be planned are from Monday to Friday, inclusively, and therefore $\mathcal{T} = \{b_1, b_2, b_3, b_4, b_5\}$ will denote the set of dates corresponding to the surgery days to be scheduled and index $t \in T = \{1, ..., 5\}$ will denote the index associated with each specific element of $\mathcal{T}$. Each OR time block within the planning horizon is then uniquely defined by a pair of indices (k, t) and will be referred to in this manner from here on.

For each patient $i \in I$, we are given the date of referral d_i, the expected duration of the surgery p_i, the urgency coefficient ρ_i, and the expected Length of Stay (LOS) μ_i, expressed in days. Let us also define function $\Phi(b_t, d_i)$ which returns the number of days elapsed between two dates b_t and d_i.

In addition, let I_j be the subset of patients that belong to specialty j, $j \in J$, and I_h the subset of patients having LOS $\mu_i = h$, $h = 1, ..., \mu_{max}$, where $\mu_{\max}$ represents the longest LOS. Clearly, subsets I_j define a partition of I as do subsets I_h.

Let O_j be the number of OR time blocks available for each specialty $j \in J$. Let us also define T_h as the subset of T corresponding to the dates in the current planning horizon $\mathcal{T}$ for which patients with $\mu_i = h$ will necessarily require a bed for the weekend if scheduled on any of those dates. For instance, for a patient having a LOS of 3, scheduling him on Monday, Tuesday or Wednesday will not require a weekend bed, but will in fact do require one if scheduled either on the Thursday or the Friday. Therefore, T_3 will include both the Thursday and Friday of the planning week considered. Note however that for values of $h \geq 6$, T_h will include all the working days of the week since scheduling a patient having such a LOS on any of those days will necessarily require a weekend stay bed. For the sake of completeness, if a scheduled patient should require hospitalization for more than one weekend this (extremely rare) situation would be treated by reducing accordingly the availability of weekend stay beds for the future planning periods affected by this decision.

Finally, we will denote by s_{kt} the time available for surgery in operating room $k \in K$ on day $t \in T$; e_{jt} the number of surgical teams available for specialty $j \in J$ on day $t \in T$; and χ the number of stay beds available during the weekend. Note that the weekend stay beds are managed as a common resource accessible to patients belonging to all surgical specialties.

3. $0-1$ linear programming formulations

In this section, we introduce $0-1$ linear programming formulations for both the SCAP and the ORPP in order to clearly highlight the hierarchy between the two decision levels present in the ORPP. These models are then used to derive a formal proof of the complexity of ORPP.
To formulate the SCAP, we assume as input data the cyclic timetable that gives, for each day of the planning horizon, the assignment of surgical specialties to OR time blocks (i.e., the master surgical schedule). Therefore, the SCAP can be considered as a particular instance of the ORPP, one in which the MSS is given. We denote this assignment with the parameter τ^j_{kt} which is equal to 1 if specialty $j \in J$ is assigned to OR time block (k, t), 0 otherwise.

Defining the following decision variables:

$$x_{ikt} = \begin{cases} 1 & \text{if patient } i \in I \text{ is assigned to OR } k \in K \text{ on day } t \in T; \\ 0 & \text{otherwise.} \end{cases}$$

the SCAP can be formulated as:

$$\min z = \sum_{i \in I, k \in K, t \in T} \Big(x_{ikt}\Phi(b_t, d_i)\rho_i + (1 - x_{ikt})(\Phi(b_5, d_i) + 1)\rho_i \Big) \tag{1}$$

$$\text{s.t.} \sum_{k \in K, t \in T} x_{ikt} \le 1 \qquad \forall i \in I \tag{2}$$

$$\sum_{i \in I_j} x_{ikt} \le M\tau^j_{kt} \qquad \forall j \in J, k \in K, t \in T \tag{3}$$

$$\sum_{i \in I} p_i x_{ikt} \le s_{kt} \qquad \forall k \in K, t \in T \tag{4}$$

$$\sum_{h=1}^{\mu_{\max}} \sum_{i \in I_h} \sum_{t \in T_h} \sum_{k \in K} x_{ikt} \le \chi \tag{5}$$

$$x_{ikt} \in \{0, 1\} \qquad \forall i \in I, k \in K, t \in T. \tag{6}$$

The objective function (1) seeks to minimize the total cost of all the patients waiting time at the end of the planning horizon as in [33]. The cost of waiting for a given patient is expressed in Need Adjusted Waiting Days (NAWDs) and is computed as the urgency coefficient of that patient at the time of planning multiplied by the elapsed waiting time of that patient since its referral date. In the objective, the first part of the inner expression computes the cost of patients included in the schedule, while the second deals with the patients that will still be waiting after the current planning period, i.e., those that are not scheduled.

Constraints (2) are the assignment constraints stating that a patient can be scheduled at most once. Constraints (3) ensure that each patient $i \in I_j$, i.e., belonging to a given specialty $j \in J$, can only be assigned to a compatible OR time block, that is one for which $\tau_{kt}^j = 1$. Note that M represents a suitably defined integer value large enough to make the constraint non binding whenever $\tau_{kt}^j = 1$. For instance M can be set to the maximum number of surgeries that could be performed in the longest OR time block across all specialties and all days of the planning horizon. For example, in a context where the shortest surgery would be 30 minutes and the longest time block 7 hours so 420 minutes, a suitable value would be $M = 14$. Constraints (4) impose that the sum of the surgery times for all the patients scheduled in any given OR time block (k, t) may not exceed that time block duration s_{kt}. The weekend stay bed availability constraint (5) ensures that the number of patients requiring a bed for the weekend is below χ, the maximum number of beds available for the weekend. Finally, constraints (6) restrict the decision variables to be binary.

To formulate the Operating Room Planning Problem (ORPP), that is the integrated model for the joint MSSP and SCAP, one needs to transform the predetermined OR time block assignment decisions τ_{kt}^j that were considered as input for the SCAP into effective decision variables. We therefore define the following family of binary variables:

$$y_{jkt} = \begin{cases} 1 & \text{if specialty } j \in J \text{ is assigned to OR } k \in K \text{ on day } t \in T; \\ 0 & \text{otherwise.} \end{cases}$$

Adding these new variables and starting from model (1)–(6), a formulation for the ORPP is obtained by replacing constraints (3) by the following four sets of contraints:

$$\sum_{i \in I_j} x_{ikt} \leq M y_{jkt} \qquad \forall j \in J, k \in K, t \in T \tag{7}$$

$$\sum_{j \in J} y_{jkt} \leq 1 \qquad \forall k \in K, t \in T \tag{8}$$

$$\sum_{k \in K, t \in T} y_{jkt} \leq O_j \qquad \forall j \in J \tag{9}$$

$$\sum_{k \in K} y_{jkt} \leq e_{jt} \qquad \forall j \in J, t \in T\,. \tag{10}$$

Constraints (7) take on the role of the τ^j_{kt} values and will force binary variable y_{jtk} to take value 1 if any patient belonging to specialty $j \in J$ is scheduled in time block (k, t) therefore assigning that specific time block to specialty j and 0 otherwise, while (8) ensure that each OR time block (k, t) is assigned to only one specialty $j \in J$. Constant M takes on the same role as in the SCAP model and is set in a similar manner. Demand constraints (9) force the number of time blocks assigned to each specialty to be equal to the number of blocks it is alloted a priori O_j (i.e., input from strategic level planning). Finally, constraints (10) limit the number of OR blocks assigned to specialty $j \in J$ on day $t \in T$ to the number of surgical teams available for that specialty on that day, e_{jt}.

The ORPP model is therefore the following:

$$\begin{aligned} \min z = & \sum_{i \in I, k \in K, t \in T} \Big(x_{ikt} \Phi(b_t, d_i) \rho_i + (1 - x_{ikt})(\Phi(b_5, d_i) + 1) \rho_i \Big) \\ \text{s.t.} \quad & (2), (4), (5), (7) - (10), (6) \\ & y_{jkt} \in \{0, 1\}\ \forall j \in J, k \in K, t \in T. \end{aligned} \tag{11}$$

As can be seen from the these formulations, there exists a clear hierarchy between decision levels and therefore between model variables: variables y (which determine the assignment of OR time blocks to surgery specialties) have a strong impact on variables x (which assign individual patients to particular OR time blocks) but not the reverse.

3.1. Complexity analysis

To the best of our knowledge, determining the complexity of ORPP has not been clearly addressed in the literature (see [23, 42]). As far as we know, no formal proof of its complexity has been published to date even though some papers, e.g., [43], have hinted that ORPP should be NP-hard given its resemblance to bin-packing, often referring to the discussion reported in [44]. The only formal proof reported is the one in [27] for the multi objective sequencing problem. Hereafter, we propose a formal proof that SCAP is NP-hard in the case of the closed block scheduling paradigm through a reduction to the $0-1$ Multiple Knapsack problem [45].

Let us consider a particular type of SCAP instance having the following characteristics: (a) the number of specialties is equal to 1, that is $J = \{1\}$, (b) the number of OR time blocks assigned to the specialty each day is equal to 1, (c) the number of stay beds is set to $\chi = \infty$. By consequence, the mathematical formulation (1)–(6) can be simplified as follows: due to assumptions (a) and (b), one can remove constraints (3), while assumption (c) makes constraint (5) unnecessary; furthermore, index k can be omitted due to assumption (b). The model (1)–(6) therefore reduces to:

$$\min z = \sum_{i \in I, t \in T} \Big(x_{it} \Phi(b_t, d_i) \rho_i + (1 - x_{it})(\Phi(b_5, d_i) + 1) \rho_i \Big)$$

$$\text{s.t.} \sum_{t \in T} x_{it} \leq 1 \qquad \forall i \in I$$

$$\sum_{i \in I} p_i x_{it} \leq s_t \qquad \forall t \in T$$

$$x_{it} \in \{0, 1\}.$$

Let us denote with $z^w = \sum_{i \in I} (\Phi(b_5, d_i) + 1) \rho_i$ the value of the worst SCAP solution, that is the solution in which no patient is scheduled for surgery. The objective function z can then be rewritten as follows:

$$z = z^w - z' \quad \text{where} \quad z' = \sum_{i \in I, t \in T} x_{it} (\Phi(b_t, d_i) + 1) \rho_i.$$

For each patient $i \in I$, z' accounts for the contribution to the objective function given by the urgency coefficient ρ_i times the number of days in the waiting list that were avoided because of the decision to schedule the patient on day $t \in T$ (i.e., setting $x_{it} = 1$). Taking advantage of the fact that z^w is a constant, we can reformulate the model as the following equivalent model:

$$\begin{aligned}
\max z' = \sum_{i \in I, t \in T} & x_{it}(\Phi(b_t, d_i) + 1)\rho_i \\
\text{s.t.} \sum_{t \in T} & x_{it} \leq 1 && \forall i \in I \\
\sum_{i \in I} & p_i x_{it} \leq s_t && \forall t \in T \\
& x_{it} \in \{0, 1\}.
\end{aligned}$$

This problem corresponds in fact to the $0-1$ Multiple Knapsack Problem (MKP) which shows that it is possible to transform any MKP instance into a particular SCAP instance. In other words, this is a reduction from MKP to SCAP. Since MKP is NP-hard [45], then SCAP is also NP-hard. The NP-hardness of ORPP therefore follows from the fact that SCAP is a particular instance of ORPP as stated earlier.

Let us point out that the use of the closed block scheduling paradigm induces a partition of the patients with respect to the surgical specialty they are assigned to. This is no longer true when planning OR blocks under the open block scheduling paradigm since any patient can then be assigned to any OR block. In other words, the open block scheduling paradigm can be seen as a setting where there is a single global specialty that includes all patients. As a consequence, the complexity proof provided here above also holds in the open block scheduling context.

4. A two level metaheuristic

The proposed solution algorithm specifically exploits the hierarchy between the two scheduling levels composing the ORPP. The main idea of the approach is to iterate the search process between the two decision levels in such a way as to address each one while trying to globally improve the current solution. The solution procedure is based on the tabu search methodology. It uses a greedy constructive heuristic to build an initial solution from which a basic search procedure is launched. This basic search explores different neighbourhoods in order to search for improved solutions until some stopping criterion is satisfied.

The overall approach is strengthened by the inclusion of an intensification procedure that is called into play whenever the basic search appears to stagnate, that is when it is unable to improve upon the best solution it has found for a predetermined number of iterations. When this situation is identified, the search process is interrupted and a specialized local search procedure is initiated from the best solution found during the basic search phase that just ended.

The robustness of the approach is further enhanced by periodically resorting to a diversification mechanism that partially deconstructs the current solution and then reconstructs a new and different one from which the search process is resumed.

The details of the different neighborhoods explored by the algorithm and its components are discussed and justified in the following subsections. The pseudo-code describing the two level metaheuristic is reported in Algorithm 1.

4.1. Neighbourhoods

As mentioned above, the approach developed here uses a variety of different neighbourhoods which together deal with the different hierarchical levels found within the ORPP structure. To deal with the patient assignment decisions, we define three different neighbourhoods. The first one, `p-swap(in,in)`, considers the exchanges of two patients that belong to two OR blocks assigned to the same specialty. The second, `p-swap(in,out)`, explores the swap of a patient included in the current schedule with one that is not. The last neighbourhood, `p-add(out,in)`, tries to add patients not currently scheduled in order to fill as much as possible the OR time blocks without exceeding their capacity and respecting their assigned surgical specialty.

With respect to the OR time block to surgical specialty assignment level, we define a fourth neighbourhood `s-swap(in,in)`, which considers the switch of surgical specialty assignments between two OR time blocks currently assigned to different specialties on different days.

Clearly, the first three neighbourhoods operate directly on the x_{ikt} decision variables and therefore do not affect the y_{jkt} variables, while the last one through its direct action on the y_{jkt} will also affect the x_{ikt} variables, which is in tune with the hierarchy of these decisions.

The complexity of `p-swap(in,in)` and `p-swap(in,out)` are quadratic in the number of patients while `p-add(out,in)` is linear. Similarly, the complexity of `s-swap(in,in)` is quadratic with respect to the number of OR time blocks.

Algorithm 1: Two level metaheuristic algorithm

Input : Empty solution $\{x_{ikt} = 0, y_{jkt} = 0\}$, problem data.
Output: The best feasible solution found for ORPP s^* and its cost z^*.

begin `/* Initialization */`
    GreedyInitialization(s_0);
    $\ell = 1$; $\ell_{NI} = 0$; isDiv $= 0$; $s^* = s_0$; $z^* = z(s_0)$; $s^*_{BS} = s_0$; $z^*_{BS} = z(s_0)$;
    while $\ell \leq N$ **do**
        if $\ell_{NI} \leq NI$ **then** `/* Basic Search */`
            s_ℓ = BestImpr($s_{\ell-1}$, p-swap(in,in), p-swap(in,out), p-add(out,in));
            $\ell_{NI} = \ell_{NI} + 1$;
            updateTabuList();
            if $z(s_\ell) < z(s^*_{BS})$ **then**
                $s^*_{BS} = s_\ell$; $z^*_{BS} = z(s_\ell)$; $\ell_{NI} = 0$;
                if $z(s_\ell) < z^*$ **then** $s^* = s_\ell$; $z^* = z(s_\ell)$;
                ;
        else `/* Intensification */`
            s_ℓ = LocalSearch($s_{\ell-1}$, s-swap(in,in)) ;
            if $z(s_\ell) < z^*$ **then** $s^* = s_\ell$; $z^* = z(s_\ell)$;
            ;
            isDiv $= 1$;
        $\ell = \ell + 1$;
        if $isDiv > 0$ **then** `/* Diversification */`
            if $isDiv = 1$ **then**
                $\bar{s}$ = p-drop($s_{\ell-1}$);
                $s_{\ell-1} = \bar{s}$;
                isDiv $= 2$;
            s_{Div} = BestImpr($s_{\ell-1}$, p-swap(in,in), p-swap(in,out)) ;
            if $z(s_{Div}) < z(s_{\ell-1})$ **then**
                $s_\ell = s_{Div}$;
                updateTabuList();
                $\ell = \ell + 1$;
                if $z(s_\ell) < z^*$ **then** $s^* = s_\ell$; $z^* = z(s_\ell)$;
                ;
            else
                isDiv $= 0$;
                $s^*_{BS} = s_{\ell-1}$;
                $z^*_{BS} = z(s_{\ell-1})$;

4.2. Greedy initialization

The basic idea of the greedy initialization procedure is to start from an empty schedule, i.e., where all x_{ikt} and y_{jkt} are set to 0, and to fill it progressively by trying to assign patients as late as possible within the week but still in time for them to be discharged from hospital without requiring a weekend stay bed. The aim of this backward filling strategy is to keep as much flexibility as possible for not yet scheduled patients by filling as tightly as possible the end portion of the planning horizon first and gradually working towards the beginning of the week where most patients can be scheduled without the need for weekend beds. Of course, the order in which patients are included in the schedule reflects their potential contribution to the societal cost function in case they are not included in the schedule, i.e., higher cost patients will therefore be considered before lower cost ones.

The procedure works as follows. Let v_i be the maximal potential contribution of patient i to the objective function (1) if not scheduled, i.e., $v_i = (\Phi(b_f, d_i) + 1)\rho_i$, where $T = \{b_1, b_2, \ldots, b_f\}$ is the set of dates of the planning horizon and b_0 the date of the day that precedes the first day in T. The algorithm first sorts all patients by increasing value of v_i such that $v_1 \geq v_2 \geq \ldots \geq v_{|I|}$. Then starting form an empty schedule, the procedure selects the first unassigned patient i in the list. Given the specialty j to which patient i belongs, the procedure then tries to insert the patient on day $t = 6 - \mu_i$, i.e., the last day of the week such that patient i may be discharged before requiring a weekend stay bed given its LOS of μ_i, in the first OR time block assigned to that specialty in which there remains enough time left to perform the surgery (line 7). If no such block exists on day t, the procedure will then check whether it is possible to assign a new OR block to specialty j on day t, i.e if there are still unassigned OR blocks on day t and the number of OR blocks already assigned to specialty j is less than its maximum value O_j (see constraints (9)). If it is, a new OR block is assigned to specialty j and patient i is assigned to that block (line 10). If the assignment of patient i to day t is not possible, the procedure then considers the previous day (line 12) and the search for a suitable OR block to insert i is then repeated for that day unless there are no more dates available within the planning horizon (i.e., $t = 0$). In this case, patient i is left unassigned, the next patient in the list is selected, and the same steps are carried out to insert this new patient in the schedule under construction. The procedure stops when all patients in the list have been treated. The pseudo-code of the greedy algorithm is reported in Algorithm 2.

Sorting the patients is $O(|I| \log |I|)$ while the patient assignment complexity is $O(|I|)$ since both the number of days t and the number k of OR time blocks assigned to a specialty j are limited by a constant value. The

Algorithm 2: Greedy initialization

Input : Empty $\{x_{ikt} = 0, y_{jkt} = 0\}$ solution, problem data.
Output: A feasible initial solution for ORPP.

1 **begin**
2   sort patients in such a way that $v_1 \geq v_2 \geq \ldots \geq v_{|I|}$;
3   **for** $i = 1, \ldots, |I|$ **do**
4     $t = b_f + 1 - \mu_i$; j = specialty of patient i; patientIsNotAssigned = true;
5   **while** $t > b_0$ *and patientIsNotAssigned* **do**
6   **if** $\exists\, y_{j,k,t} = 1$ *s.t.* $\sum_{\ell : x_{\ell k t} = 1} p_\ell + p_i \leq s_{kt}$ **then**
7         $x_{ikt} = 1$; patientIsNotAssigned = false;
8   **else**
9   **if** *OR block available and* $\sum_{k,t} y_{jkt} < O_j$ **then**
10           $y_{jkt} = x_{ikt} = 1$; patientIsNotAssigned = false;
11   **else**
12   t = previousDay(t);

overall complexity of the greedy initialization is therefore $O(|I| \log |I|)$.

4.3. Algorithm components

Basic search

The basic search (BS) consists in the selection of the best improvement among those that can be obtained by a complete exploration of the three patient-level neighbourhoods described in Section 4.1: at each iteration ℓ, we first search for the best improvement among the neighbours defined by `p-swap(in,in)` moves and store this candidate solution as s_ℓ; then, we compute the best improvement reachable with `p-swap(in,out)` moves and, if it is better than the previous value, s_ℓ is updated accordingly; and finally, these steps are repeated for `p-add(out,in)`. Once the best candidate solution within the three neighbourhoods has been identified, the procedure checks wether this new schedule improves the best solution found since the beginning of the procedure (s^*) and/or the best solution found since the start of the current phase of BS (s^*_{BS}), and updates these stored values accordingly as well as the corresponding objective function values (cf. lines 5–11).

In order to prevent the search from cycling over already visited solutions, BS uses two tabu lists having fixed lengths ℓ_1 and ℓ_2, respectively, and both implemented using tabu tags [46]. The first tabu list forbids a patient that has just been moved from being involved in any future move for the next ℓ_1

iterations while the second one forbids a patient that was just moved from an OR time block to be assigned back to the same OR block over the next ℓ_2 iterations. Obviously, it is required that $\ell_1 < \ell_2$. The rationale here is to prevent patient i that has recently been moved from moving again before the procedure has had a chance to adjust the schedule following that change. This type of tabu mechanism is particularly efficient in contexts where there is a high level of symmetry and where different sequences of moves may produce the same end result as is the case here (the reader is referred to [47, 48] for a more detailed discussion). Finally, the classical aspiration mechanism is used to lift tabu restrictions whenever the forbidden move leads to a solution better than the best solution found so far in the current BS phase.

The basic search phase is continued until a kind of stagnation criterion is met, more precisely until the number of iterations without improving upon s^*_{BS} reaches a predetermined value NI.

Intensification step

When the BS reaches the end of its current cycle, it means that the search is unable to improve the current best schedule s^*_{BS} even if at least NI patients are moved or added. We hence consider that the search has reached a sort of local minimum with respect to the patient level decisions. The algorithm will then consider moves that deal with the block assignment to specialty decisions in order to try to further improve this solution. This intensification step is done by initiating a simple local search based on the `s-swap(in,in)` neighbourhood from s^*_{BS}. Once the local search stops (i.e., when no improving neighbours can be found), the solution at hand is compared to the best solution s^* and updates are done accordingly (cf. lines 14–17).

Diversification step

In order to diversify the exploration of the solution space and thus improve the robustness of the overall solution procedure, we periodically restart the search from a new starting solution. This diversification step is performed after the end of a BS phase and the conclusion of the Intensification step that follows it.

The new solution from which to pursue the search, is obtained by applying procedure `p-drop(Solution)` to the last solution visited during the preceding BS phase, $s_{\ell-1}$. This procedure removes, from each OR time block in the schedule, the patient having the longest operating time (cf. lines 20–23). The motivation here is to free a significant amount of time in each OR block in order to facilitate a rearrangement of the remaining patients among the now partially filled blocks and the insertion of new patients in the modified schedule afterwards. A simple local search based on the `p-swap(in,in)` and `p-swap(in,out)` neighbourhoods only is then initiated from the partial schedule at hand. However, to avoid their immediate re-insertion in the solution, the patients that were just removed are declared tabu and not considered for these swaps (cf. lines 24–28).

The justification for applying this diversification step from $s_{\ell-1}$ and not to s^*_{BS} stems from the fact that $s_{\ell-1}$ is already distant from s^*_{BS} by at least NI moves. Applying the diversification step from it should therefore result in a restarting solution that is more significantly different than the one that would have been obtained from s^*_{BS}. Furthermore, when performing successive diversification steps during the course of the overall search, the restarting solutions generated will tend to be more widely dispersed than if the diversification used s^*_{BS} since this schedule might very well stay unchanged over several BS phases or cycle between a limited number of local optima.

The overall procedure then returns to the full version of the BS with this new rearranged partial schedule unless the global termination criterion has been met (cf. lines 32–34).

In the following section we will refer to the two level metaheuristic described above (Algorithm 1) as $\mathbb{A}_{\text{ORPP}}$. A reduced version of $\mathbb{A}_{\text{ORPP}}$, denoted as $\mathbb{A}_{\text{SCAP}}$, can also be defined to adapt it to the surgical case assignment problem. $\mathbb{A}_{\text{SCAP}}$ is obtained by modifying both the greedy initialization and the algorithm structure as follows. Since in the SCAP context, the master schedule is an input, i.e., the assignment of OR blocks to specialties is known and fixed, therefore the greedy initialization will only try to insert a patient in the first available OR block assigned to the corresponding specialty and will never need to assign OR blocks to specialties. With respect to the rest of the algorithm, the only difference is that the intensification phase no longer exists given the fact that the assignment of blocks to specialties is fixed.

5. Computational results

In this section we report and analyze the computational results obtained when testing algorithms $\mathbb{A}_{\text{ORPP}}$ and $\mathbb{A}_{\text{SCAP}}$. First, we describe the computational environment and the benchmark instances in Section 5.1. In Section 5.2 we briefly discuss some remarks regarding the objective function (1). The two-level approach, that is the idea of using the concept of local optimum and the use of `s-swap(in,in)` as intensification strategy are numerically tested and validated in Section 5.3 while Section 5.4 provides some insights about the characteristics that render the instances of this problem harder. Finally, Section 5.5 investigates the quality of the solutions with regards to the operating room management.

5.1. Computational environment and tuning of the parameters

$\mathbb{A}_{\text{SCAP}}$ and $\mathbb{A}_{\text{ORPP}}$ were programmed in standard C++ and compiled with `gcc 4.4.3`. For both algorithms, all tests were performed on a 1.73 GHz Intel core i7 processor, with 4 GB of RAM running under Linux Ubuntu. Computational tests with Cplex were all done with the 12.1.0 release with the default configuration and were performed on an HP ProLiant DL585 G6 server with two 2.1 GHz AMD Opteron 8425HE processors and 16 GB of RAM. Even if comparing running times is not a crucial aspect of the present analysis, it should nevertheless be noted that our algorithms were run on a machine with slightly slower CPU than the one used for Cplex.

In order to generate realistic instances on which to test our approach, we used real data provided by the Department of General Surgery of the San Martino University Hospital, Genova, Italy. In particular, this database contains all the relevant information regarding a waiting list of 400 patients: for each patient i in this list, we know the date of referral d_i, the expected duration of the surgery p_i, the urgency coefficient ρ_i, and the expected LOS μ_i.

From these informations, we generated 3 different benchmark sets, each composed of 8 instances but having a varying number of operating rooms $|K|$, length of OR block time s_{kt} and maximum number of available weekend stay beds χ. All instances however have the same number of surgical specialties $|J| = 6$ since they all were built with the same set of real surgical cases (belonging to those 6 specialties of the San Martino University Hospital) and are composed of the whole 400 patient waiting list. The characteristics of the benchmark sets are summarized in table 1.

A preliminary phase of computational experiments was carried out in order to determine adequate values for the different parameters of $\mathbb{A}_{\text{ORPP}}$. Table 2 lists these values. The same parameter settings were used for all benchmark sets, no recalibration was performed when changing set. Note that these values were the ones that provided the best average computational results in terms of solution value and running time, though on some

B_1					B_2					B_3				
Id	$\|J\|$	$\|K\|$	s_{kt}	χ	Id	$\|J\|$	$\|K\|$	s_{kt}	χ	Id	$\|J\|$	$\|K\|$	s_{kt}	χ
1	6	6	6	14	9	6	4	6	14	17	6	6	6	10
2	6	6	7	14	10	6	4	7	14	18	6	6	7	10
3	6	7	6	14	11	6	4	6	16	19	6	6	6	15
4	6	7	7	14	12	6	4	7	16	20	6	6	7	15
5	6	6	6	16	13	6	8	6	14	21	6	6	6	20
6	6	6	7	16	14	6	8	7	14	22	6	6	7	20
7	6	7	6	16	15	6	8	6	16	23	6	6	6	25
8	6	7	7	16	16	6	8	7	16	24	6	6	7	25

Table 1: Characteristics of the benchmark sets B_1–B_3.

instances slightly different parameter settings did produce better individual performances. As for algorithm $\mathbb{A}_{\text{SCAP}}$, since it is a reduced version of $\mathbb{A}_{\text{ORPP}}$, it inherits the same parameter settings as those chosen for the complete method.

Parameter	Identifier	Value
Total number of iterations	N	20000
Number of iterations w/o improvement	NI	40
Length of tabu list 1	ℓ_1	32
Length of tabu list 2	ℓ_2	38

Table 2: Parameter settings for $\mathbb{A}_{\text{ORPP}}$ and $\mathbb{A}_{\text{SCAP}}$.

5.2. Understanding the numerical behaviour of the objective function

Clearly, objective function (1) introduced in Section 3 can be separated with respect to b_0, the date of the day just preceding the first day of the schedule, as follows:

$$\begin{aligned} z &= \sum_{i,k,t} \Big(x_{ikt}\Phi(b_t, d_i)\rho_i + (1 - x_{ikt})(\Phi(b_5, d_i) + 1)\rho_i \Big) \\ &= \sum_{i,k,t} x_{ikt} \Big[\Phi(b_t, b_0)\rho_i + \Phi(b_0, d_i)\rho_i \Big] + \\ &+ \sum_{i,k,t} (1 - x_{ikt}) \Big[(\Phi(b_5, b_0) + 1)\rho_i + \Phi(b_0, d_i)\rho_i \Big]. \end{aligned}$$

It is evident, from the above reformulation, that this expression comprises an invariant part, with respect to the decision to include a patient in the schedule or not, and a variable one. The constant component is denoted as:

$$C = \sum_{i,k,t} \Big(x_{ikt}\Phi(b_0, d_i)\rho_i + (1 - x_{ikt})\Phi(b_0, d_i)\rho_i \Big) = \sum_{i,k,t} \Phi(b_0, d_i)\rho_i;$$

while the remaining variable part is expressed as:

$$V = \sum_{i,k,t} \Big(x_{ikt}\Phi(b_t, b_0)\rho_i + (1 - x_{ikt})(\Phi(b_5, b_0) + 1)\rho_i \Big).$$

V represents the social costs resulting from the choices made for the current planning period while term C accounts for those due to past planning and scheduling decisions. It would be tempting to eliminate C form the formulation since it is a constant and the decisions taken by the algorithm for the current planning period will only affect V. However, both terms need to be considered in order to adequately take into account the waiting time portion of the global societal costs. Indeed, when considering a specific patient i for inclusion in the current schedule, his portion of term C will influence the decision to schedule him/her in the current planning period by enabling a patient having a relatively low priority but that has spent a long time waiting in the list (i.e., earlier referral date d_i) to be scheduled ahead of another patient j that has a higher priority but whose waiting time is shorter (i.e., smaller value of $\Phi(b_0, d_j)$). It is through the changes in these terms when passing from one planning period to the next that the effective priority in which patients are included in the surgery schedule evolves (as well as with the reappraisal of the patients medical condition, of course). Both ways of defining the objective function are equivalent with respect to total cost, but keeping both terms induces a sort of equity with respect to the patients access to surgery.

An illustration of this phenomenon is described in table 3, where several patients belonging to the waiting list of the same specialty are considered. For each, we have computed their contribution to global societal cost as given in terms of the full objective function z and of V alone, setting the date of surgery to the last day of the current schedule ($t = b_f$) to compute both values. The patients were then ranked by decreasing values of z and V to reflect the order in which they would be considered for inclusion in the schedule (since one wants to eliminate from the list those that will contribute more in coming periods if left on it). The table reports the first 10 patients according to each criteria. As can be seen, the rankings obtained using z or V are quite different thus highlighting the importance of accounting for the patient's history in the waiting list as well as their medical priority (see patients $35-38$).

Rank	i	d_i	ρ_i	z	i	d_i	ρ_i	V
1	**6**	199	12	1872	**4**	338	45	270
2	**7**	226	12	1548	5	339	45	270
3	35	159	6	1176	3	344	45	270
4	**8**	258	12	1164	2	345	45	270
5	36	172	6	1098	1	347	45	270
6	37	193	6	972	**6**	199	12	72
7	**9**	276	12	948	**7**	226	12	72
8	38	202	6	918	**8**	258	12	72
9	**10**	290	12	780	**9**	276	12	72
10	**4**	338	45	765	**10**	290	12	72

Table 3: Comparing individual patients' contribution to z and V (values computed setting $b_f = 355$).

From a numerical standpoint, C is generally one order of magnitude larger than V (e.g., in our benchmarks, C is equal to 139 537). This of course affects the value of relative gaps and therefore must be taken into account when using such measures to compare the quality of solutions produced by any method. Consequently, in order to provide more meaningful results and to allow the reader a more complete algorithm evaluation, we provide not only the usual gaps on z but also those with respect to V.

Finally, note that the values of the worst solution, introduced in Section 3.1, can be easily computed and, for our benchmarks, are $z^w = 158\ 905$ and $V^w = 19\ 368$.

5.3. Validating the two level approach

Recall that the algorithm proposed here is based on a two level search strategy, one level focusing on the assignment of patients to OR time blocks and the other on the assignment of OR blocks to surgical specialties. Furthermore, it uses an intensification strategy – local search for the OR block assignment level – which takes as input the best solution computed during the previous basic search (BS) phase. The rationale of this intensification strategy is based on the claim that the set of best solutions computed by BS can be seen as local optima when considering only the patient assignment level (given a master surgical schedule). In other words, such solutions are good approximations of the optimal assignment of patients. Here, we provide computational results providing some proof (albeit only numerically) of the validity of this claim and of the effectiveness of the proposed intensification strategy.

Note that when one considers only the patient assignment level, one is dealing with the SCAP. Our first test therefore consists in comparing the results of $\mathbb{A}_{\text{SCAP}}$ with those obtained by using Cplex to solve model (1)–(6) using the MSS found by the procedure of [33].

For these tests, we consider the instances in B_1. We have set a time limit of 12 hours of CPU (i.e., 43200 seconds) for Cplex and $\mathbb{A}_{\text{SCAP}}$ is run using the parameters reported in table 2.

	z			V			Time (sec.)	
Id	Cplex	$\mathbb{A}_{\text{SCAP}}$	Gap	Cplex	$\mathbb{A}_{\text{SCAP}}$	Gap	Cplex	$\mathbb{A}_{\text{SCAP}}$
1	153766	153895	0.08%	14229	14358	0.91%	2063.68	27.16
2	153307	153394	0.06%	13770	13857	0.63%	43200.01	29.05
3	153343	153469	0.08%	13806	13932	0.91%	34.56	37.33
4	152854	152992	0.09%	13317	13455	1.04%	197.02	46.05
5	153761	153829	0.04%	14224	14292	0.48%	2206.34	36.74
6	153301	153433	0.09%	13764	13896	0.96%	43200.03	39.48
7	153338	153444	0.07%	13801	13907	0.77%	419.8	47.13
8	152846	153037	0.12%	13309	13500	1.44%	43200.04	37.54
			0.08%			0.89%	16815.19	37.56

Table 4: Comparison between Cplex (time limit 43200 seconds) and $\mathbb{A}_{\text{SCAP}}$ on benchmark B_1.

Table 4 reports the objective values of the best integer solutions found by Cplex and $\mathbb{A}_{\text{SCAP}}$, the relative gaps between them, considering either the z or V values and computed as $(\mathbb{A}_{\text{SCAP}}\_Best - \text{Cplex}\_Best)/\text{Cplex}\_Best$, as well as the overall running times. The last row corresponds to the averages of the corresponding columns. As can be seen, Cplex succeeds in proving optimality within the imposed time limit on only 5 out of the 8 instances of B_1. For instances 2, 6 and 8, Cplex could not completely close the gap but the solutions returned are within 0.07%, 0.07%, and 0.03% of optimality, respectively.

On the other hand, $\mathbb{A}_{\text{SCAP}}$ is able to find very good quality solutions with gaps relative to the Cplex solution of 0.89% on average with respect to the V values and of 1.44% in the worst case, all in very short times, 37.56 seconds on average, so roughly three orders of magnitude less than Cplex. These results clearly demonstrate the capability of $\mathbb{A}_{\text{SCAP}}$ to compute very good solutions fast.

In table 4, only the best local optimum found for each instance is reported of course. However, $\mathbb{A}_{\text{SCAP}}$ generates a whole sequence of local optima, each one resulting from a BS execution. To illustrate the behaviour of the search process, we have plotted in Figure 1 the values of the best solutions found by the successive BS phases with respect to the iteration count for instance 5. The figure also plots the evolution of the overall best solution found during the search (i.e., solid line). The values reported along the y-axis are the values of the V component of the objective function. Note that the origin of the y-axis refers to the optimal solution value for instance 5 as computed by Cplex.

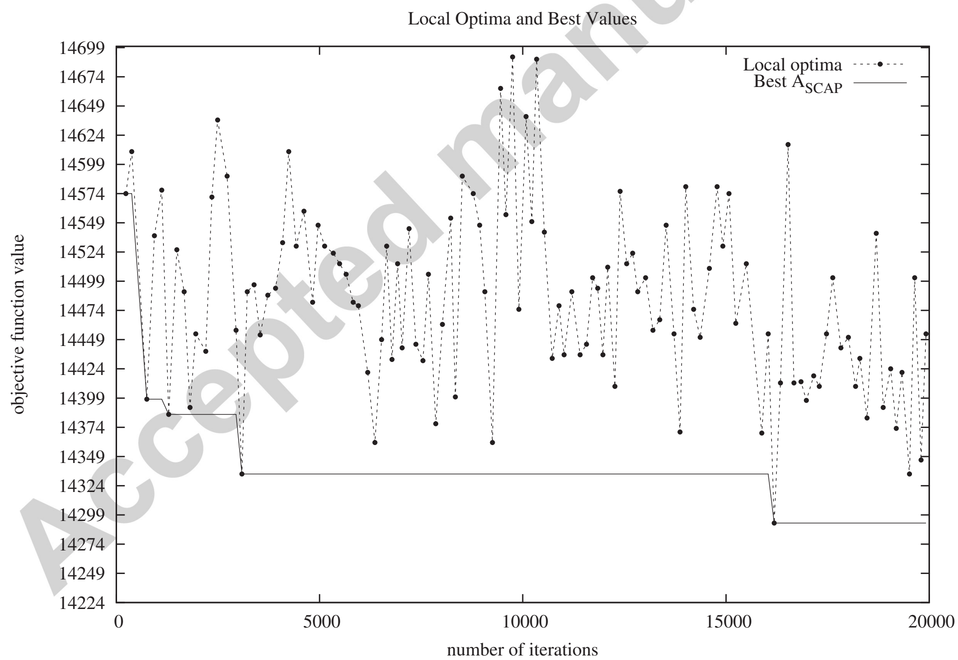

Figure 1: $\mathbb{A}_{\text{SCAP}}$: local optima and best solution (instance 5).

Figure 1 shows that the local optima have an objective value that is usually quite close to the best value returned by the algorithm in the end: hence, BS is generally able to compute good solutions whenever a master schedule is given to it. Through detailed analysis of the colletion of local optima found by the BS phases on this instance, we have computed that: 1) the gap between the average objective value across all local optima and the best one found by $\mathbb{A}_{\mathrm{SCAP}}$ was 1.03%; 2) 51.28% of the local optima encoutered have a gap lower or equal to this average gap; and 3) the worst of the local optima produced by the BS phases had a gap of 3.31%. These observations strengthen the previous conclusion regarding the efficacy of $\mathbb{A}_{\mathrm{SCAP}}$ in computing very good solutions.

	z			V				Time (sec.)	
Id	Cplex	$\mathbb{A}_{\mathrm{ORPP}}$	Gap	Cplex	Opt_G	$\mathbb{A}_{\mathrm{ORPP}}$	Gap	Best	Tot.
1	153754	153853	0.06%	14217	0.86%	14316	0.70%	2.38	31.52
2	153289	153391	0.07%	13752	0.53%	13854	0.74%	20.73	33.73
3	153315	153517	0.13%	13778	1.19%	13980	1.47%	13.52	37.10
4	152804	153088	0.19%	13267	0.72%	13551	2.14%	1.02	41.54
5	153748	153844	0.06%	14211	0.88%	14307	0.68%	10.36	29.30
6	153283	153361	0.05%	13746	0.51%	13824	0.57%	22.72	36.20
7	153311	153493	0.12%	13774	1.04%	13956	1.32%	14.77	38.69
8	152794	152947	0.10%	13257	0.79%	13410	1.15%	1.79	43.35
			0.10%		0.82%		1.10%	10.91	36.43

Table 5: Comparison between Cplex (best integer solution computed in 43200 seconds) and $\mathbb{A}_{\mathrm{ORPP}}$ on benchmark B_1, times of $\mathbb{A}_{\mathrm{ORPP}}$ in cpu seconds.

To strengthen this validation we now report in table 5 the solution values found on the same benchmark B_1 but considering now the full version of the ORPP (and not just the SCAP anymore). The table also reports for each instance the relative gaps observed between Cplex and $\mathbb{A}_{\mathrm{ORPP}}$ as well as the running times required by $\mathbb{A}_{\mathrm{ORPP}}$ to find the best solution and its overall running time. Finally, the last row reports the average values of the corresponding columns. The first thing that should be observed when solving the ORPP version of the instances of B_1 is that Cplex now fails to prove the optimality of any of the solutions it finds, reaching the 43200 seconds time limit for every instance. The objective function values reported are therefore those of the best integer solution found by Cplex. The final optimality gaps corresponding to these solutions (listed under column Opt_G) are quite small, ranging from 0.51% to 1.19% for an average of 0.82%. The time column Cplex has been omitted since it does not provide any information.

When analyzing the performance of $\mathbb{A}_{\text{ORPP}}$, one can observe that the average computing time is almost the same as that of $\mathbb{A}_{\text{SCAP}}$. Regarding the quality of the solutions found, again the analysis of the gaps between the best integer solution found by Cplex and the one found by $\mathbb{A}_{\text{ORPP}}$ clearly demonstrates the quality of the solution computed by the proposed algorithm, recording an impressive average gap of 0.1% with respect to z and of 1.1% with respect to the V component only.

Table 6 adds more details regarding the use of the two level strategy by providing the following information about benchmark B_1. The second column reports the phase of the algorithm during which the best solution was found (either the basic search BS or the intensification phase `s-swap`). Columns 3 to 6 report various measures of the improvement in solution quality as they relate to the different phases of the procedure, each time measured as a percentage of improvement and computed in a similar fashion as the gaps reported previously. More precisely, column 3 headed `s-swap`* reports the improvement resulting from the application of `s-swap` when it finds the best solution (if it effectively is `s-swap` that finds it); column 4 reports the best improvement resulting from the application of any `s-swap` intensification phase during the whole search; column 5 reports the gap between the best local optimum returned by the different BS phases and the best solution found by the complete $\mathbb{A}_{\text{ORPP}}$ algorithm (thus measuring the contribution of the intensification phases over the whole search process); and column 6 reports the total improvement achieved when comparing with the worst solution V^w (cf. section 5.2). Finally, the last 2 columns give the number of BS phases performed and the number of `s-swap` iterations done during the execution of the algorithm while the last row reports the average of the corresponding columns.

The results reported in table 6 highlight the positive impact of the proposed intensification strategy. Indeed, the idea of using `s-swap` to perform a local search at OR block level in order to improve on solutions found after using only patient based neighbourhoods is clearly justified by the fact that for every instance in B_1, the best solution was always found by an intensification phase (see column 2). In fact the presence of the `s-swap` intensification phase accounts for an additional improvement of 0.38%, on average, with respect to the best solutions obtained when using BS only (see column 5). Note also that in 2 cases out of 8 (instances 4 and 8) the best solution found is reached when `s-swap` records its best improvement. The usefulness of this intensification phase is further justified by the improvements reported

Id	Best in	Improvements on V s-swap*	s-swap	BS	V^w	BS phases	s-swap iter
1	s-swap	0.87%	1.09%	0.23%	36.23%	25	31
2	s-swap	0.71%	1.47%	0.19%	40.84%	129	95
3	s-swap	1.46%	1.95%	0.34%	40.57%	118	213
4	s-swap	1.48%	1.48%	0.46%	45.99%	124	151
5	s-swap	0.71%	2.24%	0.04%	36.29%	114	146
6	s-swap	0.90%	1.32%	0.28%	40.90%	127	109
7	s-swap	1.32%	2.35%	0.13%	40.61%	113	169
8	s-swap	2.15%	2.15%	1.39%	46.10%	119	152
		1.20%	1.76%	0.38%	40.94%	108.6	133.3

Table 6: Details about $\mathbb{A}_{\text{ORPP}}$ execution on benchmark B_1.

in columns 3 and 4, i.e., 1.20% on average when computing the best solution found, and 1.76% on average for the largest improvement generated by the different intensification phases over the overall search process.

Note that column 6, under the subheading V^w, gives the gaps between the value of the trivially bad solution V^w and the best solution found by Cplex. This measure gives an idea of the range of possible values of V over which the solution returned by $\mathbb{A}_{\text{ORPP}}$ could vary. The fact that the remaining gaps of the solutions obtained are extremely small illustrates that most of the possible improvement has already been achieved and that additional gains would probably be quite hard to get.

Finally, the benefits of the proposed approach are also highlighted by the comparison of the results reported in [33] with those appearing in table 5. In that paper, the instances tackled were significantly smaller than the ones studied here, with the larger ones having 200 patients and 3 ORs, but they were based on the same 400 patient database used here. The best solutions computed in that work, in the most favourable case, had an average gap on z equal to 0.52% (as opposed to 0.10% here) whith average running times ranging between 30 and 37 seconds. Even if these results cannot be compared directly, there seems to be a significant improvement in solution quality between the two approaches both in terms of average gap and running times. This impression will be confirmed by the analyses that follow.

5.4. Characterizing harder to solve instances

Following various discussions with OR managers working in different administrative contexts and given our own analysis of the problem, we decided to study the impact of the number of ORs and of the availability of weekend beds on the difficulty to solve ORPP instances since these two characteristics seemed to be crucial in several of the environments we knew. Benchmarks B_2 and B_3 were generated for this purpose.

Let us recall that the instances in B_2 differ from those in B_1 by the number of operating rooms available with $|K| = 4$ and 8 instead of $|K| = 6$ and 7. As for the instances in B_3, the number of weekend beds available changes from $\chi \in \{14, 16\}$ to $\chi \in \{10, 15, 20, 25\}$ while the number of operating rooms available is fixed to $|K| = 6$ (see table 1).

	z			V				Time (sec.)	
Id	Cplex	$\mathbb{A}_{\text{ORPP}}$	Gap	Cplex	Opt_G	$\mathbb{A}_{\text{ORPP}}$	Gap	Best	Tot.
9	154774	154879	0.07%	15237	0.61%	15342	0.69%	4.06	24.16
10	154399	154555	0.10%	14862	0.20%	15018	1.05%	4.95	24.65
11	154774	154867	0.06%	15237	0.70%	15330	0.61%	15.9	28.77
12	154399	154495	0.06%	14862	0.11%	14958	0.65%	11.9	31.16
13	152906	153154	0.16%	13369	1.24%	13617	1.86%	8.25	47.63
14	152369	152581	0.14%	12832	0.92%	13044	1.65%	22.61	52.1
15	152900	153151	0.16%	13363	1.27%	13614	1.88%	24.67	49.84
16	152359	152602	0.16%	12822	0.86%	13065	1.90%	39.78	52.79
			0.11%		0.74%		1.28%	16.52	38.89

Table 7: Comparison between Cplex (best integer solution computed in 43200 seconds) and $\mathbb{A}_{\text{ORPP}}$ on benchmark B_2, times of $\mathbb{A}_{\text{ORPP}}$ in cpu seconds.

Tables 7 and 8 are organized like table 5. They provide for each instance in benchmarks B_2 and B_3 respectively, the objective function value of the best solutions found by Cplex and $\mathbb{A}_{\text{ORPP}}$ in terms of z and V values, as well as the relative gaps between them. The tables also report the time at which $\mathbb{A}_{\text{ORPP}}$ found the best solution and it's total running time. As was the case with B_1, Cplex could not solve optimally any of these new instances within the 12 hours time limit alotted. The solutions reported are therefore the best integer solutions found by Cplex and the final optimality gap associated with them is given under the heading Opt_G. Finally, the bottom row reports the average of the corresponding columns.

	z			V				Time (sec.)	
Id	Cplex	$\mathbb{A}_{\mathrm{ORPP}}$	Gap	Cplex	Opt_G	$\mathbb{A}_{\mathrm{ORPP}}$	Gap	Best	Tot.
17	153778	153853	0.05%	14241	0.94%	14316	0.53%	28.06	38.81
18	153305	153430	0.08%	13768	0.57%	13893	0.91%	38.51	43.63
19	153752	153854	0.07%	14215	0.87%	14317	0.72%	24.23	36.09
20	153287	153391	0.07%	13750	0.57%	13854	0.76%	45.79	47.30
21	153742	153874	0.09%	14205	0.81%	14337	0.93%	2.98	49.21
22	153277	153364	0.06%	13740	0.58%	13827	0.63%	17.15	48.45
23	153742	153827	0.06%	14205	0.72%	14290	0.60%	23.28	48.11
24	153271	153325	0.04%	13734	0.30%	13788	0.39%	10.28	53.64
			0.06%		0.67%		0.68%	23.79	45.66

Table 8: Comparison between Cplex (best integer solution computed in 43200 seconds) and $\mathbb{A}_{\mathrm{ORPP}}$ on benchmark B_3, times of $\mathbb{A}_{\mathrm{ORPP}}$ in cpu seconds.

Results for B_3, as reported in table 8, show that the number of weekend beds available does not seem to significantly affect the hardness of the instances: the average gaps obtained here are smaller than those for B_1 however running times are larger, but all in all the differences are rather limited and do not seem to indicate a significant change in difficulty.

With respect to B_2, the results in table 7 are more interesting since they seem to show that the hardness of the instances increases somewhat as the number of operating rooms increases. Recall that instances 9 to 12 have 4 operating rooms while instances 13 to 16 have 8. The former have smaller gaps than their equivalents in B_1, with an average gap of 0.75% as opposed to 1.26%, while the latter have larger ones with an average gap of 1.82% as opposed to 0.93%. As for running times, one can observe a similar behaviour, with smaller values of both the time to find the best solution and the total time for the first four instances of B_2 when compared to the corresponding ones in B_1 (i.e., averages of 9.20 and 27.19 seconds, respectively, compared to 9.41 and 35.97) and longer times in both cases for the secong group of four instances (i.e., averages of 23.83 seconds for time to best solution and 50.59 seconds total running time compared to 12.41 and 36.89, respectively). Although the increase in difficulty as the number of ORs grows seems clear from these results, the performance of $\mathbb{A}_{\mathrm{ORPP}}$ does not seem to be jeopardized by it and remains very convincing over all instances solved.

Finally, table 9 provides additional details about the usefulness of the `s-swap` intensification phase for both benchmarks B_2 and B_3. For each of

		Improvements on V					Improvements on V		
Id	Best in	`s-swap*`	`s-swap`	`BS`	Id	Best in	`s-swap*`	`s-swap`	`BS`
9	`s-swap`	1.69%	2.20%	0.65%	17	`BS`	–	2.25%	0.00%
10	`s-swap`	0.75%	0.99%	0.14%	18	`s-swap`	0.90%	1.50%	0.13%
11	`s-swap`	1.54%	2.51%	0.76%	19	`s-swap`	0.89%	1.88%	0.08%
12	`s-swap`	0.53%	1.57%	0.36%	20	`s-swap`	0.90%	2.03%	0.19%
13	`s-swap`	0.26%	1.71%	0.04%	21	`s-swap`	0.79%	1.74%	0.38%
14	`BS`	–	0.91%	0.00%	22	`s-swap`	0.69%	1.26%	0.09%
15	`s-swap`	1.64%	1.64%	0.09%	23	`s-swap`	0.78%	2.24%	0.09%
16	`BS`	–	1.35%	0.00%	24	`s-swap`	0.73%	1.56%	0.11%
		1.07%	1.61%	0.34%			0.81%	1.81%	0.15%

Table 9: Details about $\mathbb{A}_{\text{ORPP}}$ execution on benchmarks B_2 and B_3.

them, we list: the instance "Id" number; the phase of the algorithm during which the best solution was found; the improvement resulting from the application of `s-swap` when it finds the best solution under heading `s-swap`*; and the best improvement resulting from the application of any `s-swap` intensification phase during the whole search under heading `s-swap`.

As can be seen from this table, in 13 out of 16 cases, the best solution is found through the use of the `s-swap` intensification phase. As was the case for benchmark B_1, the improvement produced by the intensification phases is significant and confirms the positive impact of using the local search at the OR block level as intensification strategy as was previously discussed.

5.5. Operating room management considerations

In the following, we analyze the solutions produced by $\mathbb{A}_{\text{ORPP}}$ from an operating room management perspective by considering the total number of patients planned for surgery according to the schedules produced, the average utilization rate of the operating rooms resulting from those schedules, and the number of post surgery beds required over the planning week.

We first report in table 10, for each instance over the three benchmarks, the number of surgery patients scheduled and the average OR utilization rates. As can be seen form the table, the schedules produced by $\mathbb{A}_{\text{ORPP}}$ exhibit very reasonable utilization rates being always higher than 90%, apart from instance 16 which is just barely under that level at 89.82%. In fact, for 17 instances out of the 24 solved, the utilization rate is higher than 95% with a 100% for instance 11, which shows that the schedules produced are using

B_1			B_2			B_3		
Id	Patients	% Util.	Id	Patients	% Util.	Id	Patients	% Util.
1	83	97.22%	9	57	97.08%	17	85	99.72%
2	92	97.38%	10	66	97.86%	18	92	96.90%
3	89	92.86%	11	58	100.00%	19	83	98.61%
4	101	93.27%	12	66	98.93%	20	92	98.57%
5	83	96.94%	13	101	91.88%	21	81	98.06%
6	94	98.81%	14	110	90.36%	22	93	98.10%
7	92	95.00%	15	99	92.08%	23	82	97.50%
8	101	93.67%	16	108	89.82%	24	92	98.10%
	91.88	95.64%		83.13	94.75%		87.50	98.19%

Table 10: Number of patients operated and utilization rate, benchmarks B_1–B_3.

efficiently the surgical capacity with respect to the available surgery time. This is confirmed by the average values for each benchmark which stand at approximately 95% for B_1 and B_2, and just over 98% for B_3. Note however, that the utilization rate is somewhat lower for the instances that have the largest number of operating rooms available (i.e., instances 13–16).

The number of scheduled patients during the planning week represents approximately 25% of the patients on the waiting list. When concentrating on the instances in B_1 and B_2, one can see that the number of patients is fairly proportional to the number of operating rooms $|K|$ available and the length of OR blocks s_{kt}. Indeed, recalling that odd numbered instances in all three benchmarks correspond to OR block durations of $s_{kt} = 6$ hours while even numbered ones have $s_{kt} = 7$ hours, one can clearly see that the former present approximately 1/7 less patients than the latter. Likewise, when only 4 ORs are available (i.e., instances 9–12 of B_2) the number of patients is almost cut in half when compared to the equivalent instances having 8 ORs (i.e., instances 13–16). Recalling that benchmark B_3 is characterized by an increasing number of stay beds available during the weekend $\chi \in \{10, 15, 20, 25\}$ with each consecutive pair of instances taking these values in order (see table 1) and considering the previous remark, the results of table 10 show that the number of scheduled patients is almost the same even when the number of weekend stay beds increases. Indeed, for the odd numbered instances this value stays between 81 and 85 patients whatever the value of χ, while for the even numbered instances it stays between 92 and 93 patients. This seems to indicate that, at least in the specific context of our partner hospital, the bottleneck of the admission process for surgery resides with the surgery time availability (in terms of number of operating room available and OR block duration) and not really with the number of weekend beds available.

We evaluated the maximum overtime required resulting from the solution obtained through a Monte Carlo Simulation over a set of 300000 scenarios for each instance in B_1. These computational results showed that the maximum overtime required was 22.05 minutes over all scenarios of all instances. In our setting the minimum duration of a surgery is 90 minutes. Considering the case in which overtime is not available then the best strategy would be to postpone the shortest surgery which would result in an utilization rate of 82.16% for the specific overtime value reported here above. This is in line with the results presented in the literature (see, e.g., [49]). Similar results are obtained for benchmark B_2 and B_3.

	weekend stay	planning horizon					weekend	
id	beds availability	1	2	3	4	5	6	7
17	10	17	28	38	42	42	10	8
18	10	19	33	38	44	47	10	5
19	15	18	29	37	42	42	15	11
20	15	18	30	40	50	44	15	12
21	20	18	27	35	40	36	20	15
22	20	19	30	41	47	51	20	13
23	25	17	28	36	32	40	25	20
24	25	18	32	40	47	48	24	18
		18.0	29.7	38.1	43.0	43.8	17.4	12.8

Table 11: Number of occupied beds for each day of the planning horizon and the weekend, benchmark B_3.

In table 11, the impact of increasing the number of weekend stay beds is analyzed in more detail for the instances in B_3. We report the number of beds required by the surgery schedule for each day of the planning horizon including the weekend (columns $3-9$) with the bottom row showing the average of the corresponding column. One can observe that weekend stay beds have a high level of utilization even when their number is increased. Indeed, for the first day of the weekend (i.e., day 6), in every instance the full quantity of weekend beds available is used, except for instance 24 in which one of the 25 beds available is not. Even for the second day of the weekend, the average utilization remains very high with 12.8 days with respect to a maximum possible value of 17.5. This shows that the algorithm really does exploit the weekend stay bed availability when selecting the patients to be operated on in order to try to further minimize the objective function.

Finally, one can also observe that the bed utilization over the days of the week resulting from the surgery schedules produced by $\mathbb{A}_{\text{ORPP}}$ does not seem balanced: the average utilization increases as the week proceeds. However, no strong conclusion should be drawn from this since the schedules are built for one week planning horizons and therefore, in the context of a rolling horizon planning approach, the patients that need to stay over the weekend after their surgery will likely increase the number of beds required at the beginning of the next week. Whether this would result in a relatively balanced usage of beds over the days of the week remains to be seen. However, even if this aspect of the problem was not considered in this study, we want to stress that the topic of bed leveling and, more generally of workload balancing, is a very challenging aspect of operating room management (see, e.g., [13]).

6. Conclusion

In this paper we have presented a two level metaheuristic algorithm that solves the joint master surgical schedule and advance scheduling problem taking into account many resource and operative constraints while minimizing the total social cost of the resulting surgery schedule.

The proposed solution approach exploits the inherent hierarchy between the two decision levels present within the problem, i.e., the assignment of OR time blocks to surgical specialties and the assignment of patients to OR time blocks. Following this hierarchical approach, $0-1$ linear programming models were introduced and exploited in order to prove that the problem is NP-hard.

The algorithm was numerically tested and validated using real data collected at the Department of General Surgery of a public hospital located in Genova (Italy). Results show that the proposed method exhibits very good performances both in terms of solution quality and computational times. The intensification strategy included in the algorithm, which is based on a local search at the OR block level, is an essential ingredient of the overall method since it was shown to contribute directly to the identification of the best solution in 21 out of the 24 instances solved. Furthermore, the proposed algorithm records an impressive average gap over the three sets of benchmark instances tested, ranging from 0.06% to 0.11%. The average gap computed on the variable component of the objective function only is likewise very impressive ranging from 0.68% to 1.28% over the three benchmark sets. Finally, from an OR management point of view, the schedules produced by the proposed algorithm exhibit very good qualities in terms of number of patients scheduled for surgery, operating room utilization rates and number of stay beds used during the weekend.

Future research avenues could consider the evaluation of different objective functions in order to directly include hospital costs. Another avenue could be to include considerations regarding the balancing of post surgery bed utilization. Finally, extentions of the modeling and solution approaches could be explored in order to deal with the uncertainty of surgery durations, for instance by including the sequencing of patients within OR time blocks, and that of patients' LOS.

Acknowledgements

All the authors acknowledge support from the Italian Ministry of Education, University and Research (MIUR), under the grant n. RBFR08IKSB, "Firb – Futuro in Ricerca 2008". Patrick Soriano's research on this project was also supported by the Natural Sciences and Engineering Research Council of Canada and the Fonds québécois de la recherche sur la nature et les technologies under grants OGP0218028, OGP0184121, and 01-ER-3254. This support is hereby gratefully acknowledged. The authors wish to thank Bernardetta Addis and Andrea Grosso for fruitful discussions. Finally, the authors wish to thank the anonymous referees for their valuable comments.

[1] Oudhoff J, Timmermans D, Knol D, Bijnen A, van der Wal G. Waiting for elective general surgery: impact on health related quality of life and psychosocial consequences. BMC Public Health 2007;7(164).

[2] Cardoen B, Demeulemeester E, Beliën J. Operating room planning and scheduling: A literature review. European Journal of Operational Research 2010;201:921–32.

[3] Guerriero F, Guido R. Operational research in the management of the operating theatre: a survey. Health Care Management Science 2011;14(1):89–114.

[4] van Oostrum J, Bredenhoff E, Hans E. Suitability and managerial implications of a master surgical scheduling approach. Annals of Operations Research 2010;178(1):91–104.

[5] Testi A, Tànfani E, Torre G. A three-phase approach for operating theatre schedules. Health Care Management Science 2007;10:163–72.

[6] Blake J, Carter M. A goal programming approach to strategic resource allocation in acute care hospitals. European Journal of Operational Research 2002;140:541–61.

[7] Dexter F, Macario A. Changing allocations of operating room time from a system based on historical utilization to one where the aim is to schedule as many surgical cases as possible. Anesthesia & Analgesia 2002;94:1272–9.

[8] Dexter F, Ledolter J, Wachtel R. Tactical decision making for selective expansion of operating room resources incorporating financial criteria and uncertainty in sub-specialties future workloads. Anesthesia & Analgesia 2005;100:1425–32.

[9] Wachtel R, Dexter F. Tactical increases in operating room block time for capacity planning should not be based on utilization. Anesthesia & Analgesia 2008;106:215–26.

[10] Zhang B, Murali P, Dessouky M, Belson D. A mixed integer programming approach for allocating operating room capacity. Journal of the Operational Research Society 2009;60:663–73.

[11] Blake J, Dexter F, Donald J. Operating room manager's use of integer programming for assigning block time to surgical groups: A case study. Anesthesia & Analgesia 2002;94:143–8.

[12] Blake J, Donald J. Mount sinai hospital uses integer programming to allocate operating room time. Interfaces 2002;32:63–73.

[13] Beliën J, Demeulemeester E, Cardoen B. Building cyclic master surgery schedules with levelled resulting bed occupancy: A case study. European Journal of Operational Research 2007;176:1185–204.

[14] van Oostrum J, van Houdenhoven M, Hurink J, Hans E, Wullink G, Kazemier G. A master surgical scheduling approach for cyclic scheduling in operating room departments. OR Spectrum 2008;30:355–74.

[15] Adan I, Bekkers J, Dellaert N, Vissers J, Yu X. Patient mix optimisation and stochastic resource requirements: A case study in cardiothoracic surgery planning. Health Care Management Science 2009;12:129–41.

[16] Banditori C, Cappanera P, Visintin F. A combined optimization-simulation approach to the master surgical scheduling problem. IMA Journal of Management Mathematics 2013;24:155–87.

[17] Choi S, Wilhelm W. On capacity allocation for operating rooms. Computers & Operations Research 2014;44:174–84.

[18] Magerlein J, Martin J. Surgical demand scheduling: A review. Health Services Research 1978;13:418–33.

[19] Ozkarahan I. Allocation of surgeries to operating rooms using goal programming. Journal of Medical Systems 2000;24(6):339–78.

[20] Guinet A, Chaabane S. Operating theatre planning. International Journal of Production Economics 2003;85:69–81.

[21] Lamiri M, Xie X, Dolgui A, Grimaud F. A stochastic model for operating room planning with elective and emergency demand for surgery. Journal of the Operational Research Society 2008;185:1026–37.

[22] Fei H, Chu C, Meskens N, Artiba A. Solving surgical cases assignment problem by a branch-and-price approach. International Journal of Production Economics 2008;112:96–108.

[23] Hans E, Wullink G, van Houdenhoven M, Kamezier G. Robust surgery loading. European Journal of Operational Research 2008;185:1038–50.

[24] Marques I, Captivo M, Pato M. An integer programming approach to elective surgery scheduling. OR Spectrum 2012;34:407–27.

[25] Rizk C, Arnaout J. ACO for the surgical cases assignment problem. Journal of Medical Systems 2012;36:1191–9.

[26] Pham D, Klinkert A. Surgical case scheduling as a generalized job shop scheduling problem. European Journal of Operational Research 2008;185:1011–25.

[27] Cardoen B, Demeulemeester E, Beliën J. Optimizing a multiple objective surgical case sequencing problem. International Journal of Production Economics 2009;119:354–66.

[28] Roland B, Martinelly C, Riane F, Pochet Y. Scheduling an operating theatre under human resource constraints. Computers & Industrial Engineering 2010;58:212–20.

[29] Riise A, Burke E. Local search for the surgery admission planning problem. Journal of Heuristics 2011;17(4):389–414.

[30] Herring W, Herrmann J. The single-day surgery scheduling problem: sequential decision-making and threshold-based heuristics. OR Spectrum 2012;34:429–59.

[31] Meskens N, Duvivier D, Hanset A. Multi-objective operating room scheduling considering desiderata of the surgical teams. Decision Support Systems 2013;55:650–9.

[32] Jebali A, Alouane A, Ladet P, Roland C. Operating rooms scheduling. International Journal of Production Economics 2006;99:52–62.

[33] Tànfani E, Testi A. A pre-assignment heuristic algorithm for the master surgical schedule problem (MSSP). Annals of Operations Research 2010;178(1):105–19.

[34] Choi S, Wilhelm W. An approach to optimize block surgical schedules. European Journal of Operational Research 2014;235:138–48.

[35] Agnetis A, Coppi A, Corsini M, Dellino G, Meloni C, Pranzo M. A decomposition approach for the combined master surgical schedule and surgical case assignment problems. Health Care Management Science 2014;17:49–59.

[36] Aringhieri R, Tànfani E, Testi A. Operations Research for Health Care Delivery. Computers & Operations Research 2013;40(9):2165–6.

[37] Glover F, Laguna M. Tabu Search. Kluwer Academic Publishers, Norwell, MA; 1997.

[38] Gendreau M. Recent advances in tabu search. In: Ribeiro C, Hansen P, editors. Essays and Surveys in Metaheuristics. Kluwer Academic Publishers; 2002, p. 369–77.

[39] Testi A, Tànfani E, Valente R, Ansaldo G, Torre G. Prioritising surgical waiting list. Journal of Evaluation in Clinical Practice 2008;14(1):59–64.

[40] Valente R, Testi A, Tanfani E, Fato M, Porro I, Santori G, et al. A model to prioritize access to elective surgery on the base of clinical urgency and waiting time. BMC Health Services Research 2009;9(1).

[41] Min D, Yih Y. An elective surgery scheduling problem considering patient priority. Computers & Operations Research 2010;37:1091–9.

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

[43] Fei H, Meskens N, Chu C. A planning and scheduling problem for an operating theatre using an open scheduling strategy. Computers & Industrial Engineering 2010;58:221–30.

[44] Houdenhoven MV, Oostrum JV, Hans E, Wullink G, Kazamier G. Improving operating room efficiency by applying bin-packing and portfolio techniques to surgical case scheduling. Anesthesia & Analgesia 2007;105(3):707–14.

[45] Martello S, Toth P. Knapsack Problems: Algorithms and Computer Implementations. Wiley-Intersci. Ser. Discrete Math. Optim., John Wiley and Sons; 1990.

[46] Gendreau M, Hertz A, Laporte G. A tabu search heuristic for the vehicle routing problem. Management Science 1994;40(10):1276–90.

[47] Aringhieri R, Dell'Amico M. Comparing metaheuristic algorithms for sonet network design problems. Journal of Heuristics 2005;11(1):35–57.

[48] Aringhieri R. Composing medical crews with equity and efficiency. Central European Journal of Operations Research 2009;17(3):343–57.

[49] Tyler D, Pasquariello C, Chen C. Determining optimum operating room utilization. Anesthesia & Analgesia 2003;96(4):1114–21.