KATHOLIEKE UNIVERSITEIT
LEUVEN

Faculty of Economics and Applied Economics

Optimizing a multiple objective surgical case scheduling problem

Brecht Cardoen, Erik Demeulemeester and Jeroen Beliën

DEPARTMENT OF DECISION SCIENCES AND INFORMATION MANAGEMENT (KBI)

KBI 0625

Optimizing a multiple objective surgical case scheduling problem

Brecht Cardoen, Erik Demeulemeester, Jeroen Beliën

Katholieke Universiteit Leuven, Faculty of Economics and Applied Economics, Department of Decision Sciences and Information Management, Naamsestraat 69, B-3000 Leuven, Belgium, brecht.cardoen@econ.kuleuven.be, erik.demeulemeester@econ.kuleuven.be, jeroen.belien@econ.kuleuven.be

Abstract

The scheduling of the operating theater on a daily base is a complicated task and is mainly based on the experience of the human planner. This, however, does not mean that this task can be seen as unimportant since the schedule of individual surgeries influences a medical department as a whole. Based on practical suggestions of the planner and on real-life constraints, we will formulate a multiple objective optimization model in order to facilitate this decision process. We will show that this optimization problem is NP-hard and hence hard to solve. Both exact and heuristic algorithms, based on integer programming and on implicit enumeration (branch-and-bound), will be introduced. These solution approaches will be thoroughly tested on a realistic test set using data of the surgical day-care center at the university hospital Gasthuisberg in Leuven (Belgium). Finally, results will be analyzed and conclusions will be formulated.

Keywords: health care, scheduling, integer programming, branch-and-bound

---

1 Introduction

Health care is omnipresent in today's developed world and has evolved towards a major business industry characterized by increasing competition. This means that health care managers, comparable to their industrial counterparts, continually seek to improve the quality of their services and to reduce operational costs. In order to do so, the field of operations research and operations management should provide interesting insights (Carter 2002). When we take a closer look at the operational facilities of a hospital center, we can identify the operating theater as a major cost driver. Next to the inherent complexity of surgeries and the costs of the operating room itself, the linking aspect of the operating theater to other facilities contributes to its importance. Beliën and Demeulemeester (2006) clearly indicate that the quality of the nurse schedule can be improved by adapting the surgery schedule. Next to nurse scheduling, the operating theater interacts, for example, with the instrument sterilization facility or the ward planning. The

central role of the surgery scheduling process in a medical setting makes it an interesting and promising subject for improvement identification and will consequently constitute the focus of this research paper.

In the literature, the surgery scheduling process for elective cases is often seen as a three stage process (Blake and Donald 2002, Beliën and Demeulemeester 2007). In a first stage, one has to determine how much operating room time is assigned to the different surgeons or surgical groups. This stage is often referred to as case mix planning and is situated on a strategic level (Blake and Carter 2002). The second stage, which is tactically oriented, concerns the development of a master surgery schedule. This schedule can be seen as a cyclic timetable that defines the number and type of operating rooms available, the hours that rooms will be open, and the surgeons or surgical groups to whom the operating room time is assigned (Blake, Dexter and Donald 2002). In the third and final stage, individual patients or cases can be scheduled on a daily base. It is on this operational level that our research should be situated.

Methodologies for scheduling individual surgical cases are often based on a two-step procedure. In a first step, surgeries are assigned to the operating rooms. The second step consists of sequencing the surgeries within each operating room. Jebali, Alouane and Ladet (2006) developed a solution procedure based on this distinction. In the assignment step, they try to minimize overtime, undertime and patient waiting time (between surgery and hospitalization day), whereas the objective in the sequencing step is limited to overtime minimization. Both objective functions are formulated in terms of costs and are optimized using a mixed integer programming approach. A similar two-step procedure can be found in Guinet and Chaabane (2003), though their focus lies primarily in the assignment phase. Using a primal-dual heuristic, they try to optimize the patient waiting time and the operating theater overload. Both papers link the operating theater to a single recovery room. Sier, Tobin and McGurk (1997) described the sequencing step as a mixed integer nonlinear programming formulation and developed a simulated annealing heuristic in order to optimize their multi-objective function. The sequencing step was also the subject of research by Hsu, de Matta and Lee (2003). They introduced a tabu search-based heuristic in order to minimize the number of nurses in the single postanesthesia care unit and the completion time of the last patient in that unit. Similarly to our research, their model is developed for an ambulatory surgical center. A different two-stage approach can be identified in Marcon, Kharraja and Simonnet (2003). In order to master the risk of no

realization of surgeries and the utilization stabilization of the operating rooms, they make a distinction between a static and a dynamic phase. During the static phase, a multiple knapsack problem is solved in order to get to a fixed schedule. They state, however, that the execution of this schedule during the surgery day will be influenced by unforeseen events. The monitoring and rescheduling due to these events is done in the dynamic phase. Both integer programming and simulation are used to evaluate their procedure. Other approaches for the detailed planning of elective cases can, for instance, be found in Weiss (1990) or Ozkarahan (2000). In Section 2, we will highlight the two phases that structure this research.

An important issue in the field of operations research and operations management is the applicability of the research. Ideally, theoretical insights should be extended to the society and should be practically implemented. However, it seems that the developed techniques are still not widely adapted in a medical setting. Recently, Sainfort et al. (2005) discovered that there is only little planning at a systemic level in terms of patient flow, capacity planning or resource allocation amongst the European countries. For Flanders, in particular, they even did not find examples of current research studies dealing with the issues above. Based on the considerations mentioned, we want to reverse the relation between theory and practice in this research. This implies that we maintain a close cooperation with the surgical day-care center at Gasthuisberg, where our study originated, and try to apply and develop techniques for their real-life decision problem.

The paper is organized as follows. In Section 2, we will extensively describe the objectives, constraints and characteristics of the *surgical case scheduling problem* (SCSP) of interest. The next two sections are devoted to the development of solution approaches. A pure integer programming (IP) model is introduced and consecutively enhanced in Section 3, whereas Section 4 describes both exact and heuristic branch-and-bound (B&B) procedures. A test set is introduced in Section 5 and computational results will be summarized and evaluated. Finally, in Section 6, conclusions will be formulated and ideas for future research will be mentioned.

2 Problem statement

The problem we will investigate emerged from the suggestions and needs of the day-care center at Gasthuisberg. Today, the general scheduling process of surgeries is as follows. When patients

have a request for surgery, they are assigned to a certain surgery day with spare capacity. Since the final surgery schedule is made only one day in advance, patients are unaware of the time they should enter the hospital until the evening before their surgery day. At this point, patients should call the hospital in order to get their expected arrival time. Registration in the hospital happens one hour before surgery. When a surgery is almost finished, the next patient is transferred to a pre-surgical treatment room where the preparation is done. After surgery, the patient is transferred to the first recovery room (recovery phase 1) to get through the critical awakening phase. When the patient is conscious and the awakening process tends to be normal, he or she is moved to a second recovery room (recovery phase 2) where the patient stays until the surgeon gives permission to leave the hospital.

It is not so hard to distinguish the two phases in this scheduling process. In the assignment step, patients are assigned to surgery days on which their surgeon is present and there is spare capacity. When surgeons only have one surgery block a day at their disposal, this problem is equal to the assignment of patients to operating rooms. However, in the SCSP we allow that surgeons possibly have multiple surgery blocks a day in different operating rooms. Since the surgeon cannot perform different surgeries simultaneously, these blocks have to be sequential. A reason for the change in operating room can, for instance, be found in the fixed equipment available in the operating room. Since we do not know which patients will require surgery in the future, the assignment policies have to be developed in an on-line environment. Remark that the overall quality of the final surgery schedule will strongly depend on the constitution of the assigned population. In the sequencing phase, we will try to derive a good schedule for the population of a surgery day. On the one hand, we have to decide which surgery will be performed in which operating room. On the other hand, we have to find an appropriate sequence within each operating room. When we will explicitly unfold the objectives and constraints of the SCSP, it will be clear that the operating rooms interact and that they consequently cannot be sequenced independently. In this paper, we will assume that patients are already assigned to surgery days. In other words, we will only investigate the sequencing phase of the problem and take the assigned population as given. In the remainder of this section, we will specify the SCSP and try to capture the logic into a mathematical formulation. When no mathematical formulation is stated, the logic is captured in the structure of the other formulations. A binary decision variable x_{ips} will be introduced. This variable equals 1 if a surgery of type i starts

on period p by surgeon s. The symbols used throughout the equations are described in the Appendix.

2.1 Objectives

The SCSP comprises 6 objectives that have to be optimized simultaneously. In order to do so, we will capture the objectives into one multi-objective function that will be minimized.

2.1.1 Description of the objectives

The first and second objective basically follow the structure in which costs are linked to the starting time of certain types of surgeries. The underlined period, represented in Figure 1, is a time indication and is needed as a reference to calculate penalty costs. When a surgery starts before or on this limit (i.e. on period 2 or 3), no costs occur. However, when the particular surgery is performed after the reference period, the cost will be equal to the difference between the starting period of the surgery and the reference. The larger this gap, the larger the penalty, as can be seen in the cost function. Starting, for instance, a surgery of type i on period 6 brings along a penalty cost of 6-3=3.

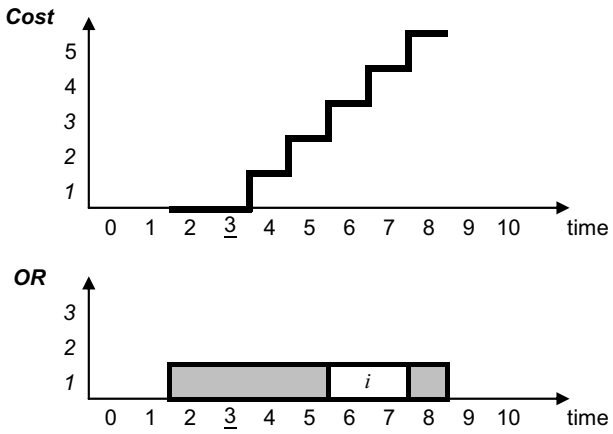

Figure 1: General illustration of objective 1 and objective 2.

A first objective concerns the scheduling of surgeries on children. For medical reasons, patients need to be sober when the surgery is performed. Contrary to adults, children cannot easily deal with this obligation and the lack of food can cause parents, surgeons or other patients a lot of annoyance. In order to avoid such a situation, it is desirable to schedule these surgeries

as early as possible. In Equation 1, α_1 represents the sum of the starting times of surgeries performed on children. Decreasing the value of α_1 would hence result in improving the surgery schedule with respect to this objective. Since the reference period for this first objective is equal to 0, the cost associated with the surgery start for a child on period p is equal to $p - 0 = p$.

$$\sum_{s \in S} \sum_{r \in R_s} \sum_{i \in I_s \cap I_r \cap I_{child}} \sum_{p=max\{P_{rs}^{lb}, presurgref \cdot \theta_i^{presurg}\}}^{P_{rs}^{ub} - k_i + 1 - \tau_{irs} \cdot k_{clean}} p \cdot x_{ips} = \alpha_1 \tag{1}$$

We already mentioned that the second objective is very similar to the first one, though this time we are concerned about prioritized patients (Equation 2). We can think, for example, of very urgent surgeries or patients who had already a cancelled surgery. Since it is possible in reality that patients get cancelled at the end of the day due to delays, we want the prioritized patients to be scheduled before a reference period. We distinguish between the children and the prioritized patients since both the reference period and the weight assigned to the objectives (see Section 2.1.2) can be different. In our case, however, we will use the same reference period for both objectives, i.e. we also want the prioritized patients to be scheduled as early as possible. In Equation 2, α_2 represents the sum of the starting times of surgeries performed on prioritized patients.

$$\sum_{s \in S} \sum_{r \in R_s} \sum_{i \in I_s \cap I_r \cap I_{prior}} \sum_{p=max\{P_{rs}^{lb}, presurgref \cdot \theta_i^{presurg}\}}^{P_{rs}^{ub} - k_i + 1 - \tau_{irs} \cdot k_{clean}} p \cdot x_{ips} = \alpha_2 \tag{2}$$

Objective 3 takes patients into account who have a large travel distance with respect to the hospital. Although the surgical day-care center of Gasthuisberg is centrally positioned in Belgium, it is possible that patients have to travel over 150 kilometers. The aim is to schedule these patients after a certain reference period. Contrary to objective 1 and objective 2, the penalty does not increase with the number of periods between the surgery start and a reference period. In other words, a patient is scheduled before the travel limit (penalty cost) or the surgery starts on or after the travel limit (no penalty cost). In Equation 3, the number of travel patients with a surgery start time $< travelref$ is counted and captured by the help variable α_3. The relevance of this objective is twofold. On the one hand, there is an increase in patient satisfaction if the effort patients have to do in order to get in time in the hospital is not too large. On the

other hand, the arrival time of travel patients is uncertain due to the larger distance. If patients have more time to manage the trip, there is less chance to have a delay in the surgery schedule.

$$nrtravelpat - \sum_{s\in S}\sum_{r\in R_s}\sum_{i\in I_s\cap I_r\cap I_{travel}}\sum_{p=max\{travelref,P^{lb}_{rs},presurgref\cdot\theta_i^{presurg}\}}^{P^{ub}_{rs}-k_i+1-\tau_{irs}\cdot k_{clean}} x_{ips} = \alpha_3 \qquad (3)$$

The current scheduling practice does not explicitly take the length of recovery into account. Therefore, it is not unlikely that surgeries that are characterized by a large recovery time in phase 1 or phase 2 are treated by the end of the day. The consequence is that these patients still need care when the day-care department closes and that they have to be hospitalized. Such unplanned hospitalization is of course very expensive and decreases patient satisfaction. We will try to tackle this problem by minimizing the number of periods in which recovery care has to be given after closing time of the surgical day-care center ($= \alpha_4$). This goal will constitute our fourth objective and is mathematically formulated in Equation 4. It should be clear that the start time of the recovery depends on the start time of the surgery.

$$\sum_{p=P^{ub}+1}^{mrp}\sum_{s\in S}\sum_{r\in R_s}\sum_{i\in I_s\cap I_r}\sum_{p'=max\{p-k_i-l_i-m_i+1,P^{lb}_{rs},presurgref\cdot\theta_i^{presurg}\}}^{min\{p,P^{ub}_{rs}-k_i+1-\tau_{irs}\cdot k_{clean}\}} x_{ip's} = \alpha_4 \qquad (4)$$

Next to the recovery overtime caused by inadequate scheduling, we also want to deal with the highly unleveled occupation pattern of both recovery phases and work towards a smooth utilization of the recovery beds. The relation between the surgery schedule and the resulting bed occupancy has amongst others been studied by Harris (1985) and Kim and Horowitz (2002). Objective 5 consists of minimizing the peak number of beds used in recovery phase 1 (α_5 in Equation 5), whereas objective 6 focuses on the peak number of beds used in the second recovery phase (α_6 in Equation 6). Note that smoothing the bed occupancy also results in a leveled workload for the nursing personnel and consequently reduces working pressure.

$$\sum_{s\in S}\sum_{r\in R_s}\sum_{i\in I_s\cap I_r}\sum_{p'=max\{p-k_i-l_i+1,P^{lb}_{rs},presurgref\cdot\theta_i^{presurg}\}}^{min\{p-k_i,P^{ub}_{rs}-k_i+1-\tau_{irs}\cdot k_{clean}\}} x_{ip's} \leq \alpha_5 \quad \forall p : min_{r,s}P^{lb}_{rs} \leq p \leq mrp \qquad (5)$$

$$\sum_{s\in S}\sum_{r\in R_s}\sum_{i\in I_s\cap I_r}\sum_{p'=max\{p-k_i-l_i-m_i+1,P_{rs}^{lb},presurgref\cdot\theta_i^{presurg}\}}^{min\{p-k_i-l_i,P_{rs}^{ub}-k_i+1-\tau_{irs}\cdot k_{clean}\}} x_{ip's} \leq \alpha_6 \quad \forall p : min_{r,s}P_{rs}^{lb} \leq p \leq mrp \tag{6}$$

2.1.2 Towards a multiple objective function

Intuitively, it seems necessary to take multiple objectives into account. When we would optimize the surgery schedule for only one objective, it would be very likely that this schedule would perform poorly with regard to some other objectives. Question is, however, how we should combine the objectives into a well-balanced multi-objective function. One possible, but somehow naive approach would be to sum the values for each objective (i.e. $\sum_{j=1}^{6}\alpha_j$) and to minimize this function. We can come up with two considerations why this objective function would end up with surgery schedules that are not very favorable for the decision maker or the planner. First, several objectives are expressed in different units. Objective 1, for instance, is expressed in periods, whereas objective 5 is defined in terms of beds. Since we will allow for 288 periods in a day (5-minute periods) and only 8 beds in recovery phase 1, the discrepancy is obvious. One possible solution to this problem is to define a trade-off between a period and a bed. This, however, is a very subjective decision, even for an experienced planner, and is difficult to argument. Second, the trade-off problem even exists for objectives that are expressed in the same unit. We cannot guarantee that the range between the worst and the best schedule for one objective is comparable to that for another objective, so that correction is possibly needed.

The multiple objective function we want to propose is based on the *room for improvement* (RFI_j) for each objective j and is unitless. Since we know the population of surgeries that has to be scheduled (including the idle periods), we can generate for each single objective, i.e. leaving all other objectives out of consideration, the best and the worst schedule. This implies for $j \in \{1,2,3,4\}$ that we have to minimize α_j in order to get the best (=smallest) value and that we should maximize α_j in order to get the worst (=largest) value. We have to change this procedure for objective $j \in \{5,6\}$ since there is a problem in finding the worst value: maximization of, for instance, α_5 will always lead to a value equal to the total capacity of the recovery room in phase 1 (=cap_l). Yet it is possible that the worst schedule of the population never needs that much capacity, in which case our worst value is not realistic. Therefore we should maximize

α_5 for each period individually and change the "$\leq$-sign" in Equation 5 that corresponds with the particular period into an "=-sign". The worst value is consequently the largest occupancy over all the periods. No modifications are needed in order to find the best schedule for objective $j \in \{5, 6\}$. Calculation of the extreme values for α_1 up to α_6 is done during instance generation (see Section 5) using the ILOG CPLEX 8.1 optimization library. Since the optimization of even a single α_j could be hard, we model the problem as a mixed integer problem in which all variables are continuous, except for the integer variable α_j to be optimized.

For objective j we can now calculate the room for improvement as given in Equation 7.

$$RFI_j = \frac{\alpha_j - bestvalue_j}{worstvalue_j - bestvalue_j} \tag{7}$$

One nice feature of this transformation is that we have a relative measure and hence do not have to struggle with units anymore: we get a value that is in the range [0,1] and is easy to interpret since the RFI_j indicates how much worse the value of objective j is with regard to its best value. If RFI_j equals 0, we cannot improve objective j any further. One could argue why we do not divide α_j by its best value and optimize this formulation since it would be a relative measure too. Answer is that the best value of an objective can possibly be equal to 0 and that we cannot divide by 0. One could argue again that when the worst and the best value are equal to each other, we also have to divide by 0. However, when this is the case, there is no need to optimize the particular objective since every feasible schedule will have the same value for that objective. In other words, we only take those objectives j into account for which $bestvalue_j \neq worstvalue_j$ and group them in set J.

Using the transformation described above, all objectives will be gradually optimized to the same level and will somehow be comparable to each other. It is unlikely, however, that the objectives are of equal importance to the human planner. Thus we should incorporate a possibility for the planner to express the relevance of the different objectives. This can easily be done by assigning weights to the different objectives. Note that these weights only indicate the preferences of the scheduler. The multiple objective function for the SCSP of the day-care center in Gasthuisberg is then in general equal to $\sum_{j \in J} w_j \cdot (\alpha_j - bestvalue_j)/(worstvalue_j - bestvalue_j)$. Note that when the sum of the weights equals 1, the multiple objective function still has a value that is in the range [0,1].

2.2 Constraints

Decisions made on the strategic and tactical level of the operating theater planning (see Section 1) definitively have their impact on the detailed planning of surgeries and should therefore be taken into account. This implies, for instance, that each surgeon is restricted to start and end his or her surgeries during the time and in the operating rooms assigned by the master surgery schedule. Recall that operating rooms can differ in opening hours and that surgeons can perform surgeries in multiple operating rooms, though not simultaneously.

It is possible that some patients still have to do some pre-surgical tests (e.g. X-ray) on the day of the surgery. Conceptually, we can add this specialty through the introduction of a reference period: surgery types $i \in I_{presurg}$ must start on or after the reference period $presurgref$ in order to create time for the patient to do the tests.

When building the surgery schedule, it is essential that for each surgeon his or her total population of patients is planned (Equation 8) and that surgeries do not overlap (Equation 9). This means that surgeries cannot start when the operating room is occupied by any other surgery. Note that idle periods are also treated as surgery types.

$$\sum_{r \in R_s} \sum_{p=max\{P_{rs}^{lb}, presurgref \cdot \theta_i^{presurg}\}}^{P_{rs}^{ub}-k_i+1-\tau_{irs} \cdot k_{clean}} x_{ips} = |N_s|_{type_n=i} \quad \forall s \in S, \forall i \in I_s \tag{8}$$

$$\sum_{i \in I_s \cap I_r} \sum_{p'=max\{p-k_i+1, P_{rs}^{lb}, presurgref \cdot \theta_i^{presurg}\}}^{min\{p, P_{rs}^{ub}-k_i+1-\tau_{irs} \cdot k_{clean}\}} x_{ip's} \leq 1 \quad \forall s \in S, \forall r \in R_s, \forall p : P_{rs}^{lb} \leq p \leq P_{rs}^{ub} \tag{9}$$

Both recovery areas are characterized by a limited capacity so that the peak to be minimized through objectives 5 and 6 cannot be larger than the total number of beds available in the respective recovery rooms (Equation 10 and 11).

$$\alpha_5 \leq cap_l \tag{10}$$

$$\alpha_6 \leq cap_m \tag{11}$$

Since instruments are needed during surgery and since their capacity is limited, problems can arise due to inadequate scheduling. We will explicitly take this difficulty into account by

adding the constraints of Equation 12 to the model. For each instrument type and period we have that the number of instruments used in that period cannot exceed the number available. The availability of the instruments does not solely depend on the simultaneous use of a type of instrument over the different operating rooms. After use, instruments possibly need to be sterilized for several periods and hence cannot be used for subsequent surgeries. The sterilization duration can differ between the types of instruments.

$$\sum_{s \in S} \sum_{r \in R_s} \sum_{i \in I_s \cap I_r \cap I_e} \sum_{p'=max\{p-k_i-ster_e+1, P_{rs}^{lb}, presurgref \cdot \theta_i^{presurg}\}}^{min\{p, P_{rs}^{ub}-k_i+1-\tau_{irs} \cdot k_{clean}\}} x_{ip's} \leq cap_e \tag{12}$$

$$\forall e \in E, \forall p : min_{r,s} P_{rs}^{lb} \leq p \leq max_{r,s} P_{rs}^{ub}$$

Finally, we also have to deal with the occurrence of infections (Equation 13). One common infection is, for instance, the notorious and dangerous hospital bacteria. After the surgery of an infected patient, the operating room normally needs additional cleaning for k_{clean} periods. This cleaning, however, is not required when the next patient has exactly the same infection. When an infected patient is the last one to be treated in an operating room, no additional cleaning needs to be performed since the entire operating theater is thoroughly cleaned at closing time. However, when an infected patient is scheduled in a surgery block that is followed by a surgery block of a different surgeon, the cleaning is obligatory and should be entirely performed in the surgery block of the infected patient. We do assume in other words that, with regard to infections, surgeons work independently. Note that we can reduce the number of constraints originating from Equation 13 by introducing a Big-M formulation. This, however, would result in a weaker formulation and hence would lead to inferior results.

$$x_{i'p's} \leq 1 - x_{ips} \tag{13}$$

$$\forall s \in S, \forall r \in R_s, \forall i \in I_s \cap I_r \cap I_{bact}$$

$$\forall p : min\{P_{rs}^{lb}, presurgref \cdot \theta_i^{presurg}\} \leq p \leq P_{rs}^{ub} - k_i + 1 - \tau_{irs} \cdot k_{clean}$$

$$\forall i' \in I_s \cap I_r \setminus \{idle\} : bact_{i'} \neq bact_i$$

$$\forall p' : max\{p, presurgref \cdot \theta_i^{presurg}\} \leq p' \leq min\{p + k_i - 1 + k_{clean}, P_{rs}^{ub} - k_{i'} + 1 - \tau_{i'rs} \cdot k_{clean}\}$$

2.3 Complexity analysis

In this section we will prove that the optimization of the SCSP is computationally hard, i.e. NP-hard, by showing that the SCSP contains a problem, for which the optimization is already shown to be NP-hard, as a special case. This technique is referred to as a proof by restriction (Garey and Johnson 1979). In particular, we will specify restrictions so that the restricted SCSP, which we will refer to as R-SCSP, will be identical to the *resource investment problem* (RIP).

Theorem 1. *Problem SCSP is NP-hard.*

Proof of Theorem 1. The RIP is situated in the domain of the *resource-constrained project scheduling problems* (RCPSP) and the optimization is shown in Neumann, Schwindt and Zimmerman (2003) to be NP-hard. In the RIP, one has to provide resources to a project such that it can be finished before the deadline. The costs associated with the peak use of each resource during the course of the project are to be minimized (Equation 14), where $cost_o$ denotes the procurement cost per unit of resource o. The optimization should incorporate both the limited availability of resources at each time instance p and the precedence relations between activities.

$$Minimize \sum_{o} cost_o \cdot max_p \, resource_{op} \tag{14}$$

In the R-SCSP, we assume that $I_{child} = I_{prior} = I_{travel} = I_{bact} = I_{presurg} = \emptyset$ and $P^{ub} > mrp$. In other words, we will only take objective 5 and 6 into account, i.e. minimizing the peak number of beds used in recovery phase 1 and 2. No bottleneck instruments are needed ($E = I_e = \emptyset$) and $S = R = R_s = \{1\}$.

We cannot straightforwardly identify the RIP in the R-SCSP since there is a problem with the activity representation. We cannot define an activity for the RIP to be equal to an entire surgical process of a patient since this process is actually a sequence of three distinct activities. First, there is the surgery itself which takes place in the operating room. Second, a recovery process is initiated in recovery phase 1. Finally, the patient is transferred for a second recovery process to recovery phase 2. The last two activities, though, consume resources when the surgery itself is already finished. This feature is atypical for the RIP and should hence be modified. Instead of scheduling a patient n with $type_n = i$ as a sequence of 3 activities, we will schedule 3 precedence related (fictive) patients, namely n', n'' and n''', each representing one activity. This substitution is depicted in Figure 2. In this figure, an activity-on-the-node representation

is introduced. The duration of the activity is indicated above the node, whereas the resource consumption is indicated below using a vector. Only three resource types are represented in the R-SCSP, i.e. the operating room ($o = 1$), beds of recovery phase 1 ($o = 2$) and beds of recovery phase 2 ($o = 3$). The consumption of these resources by each activity is indicated in the respective entries of the vector: $\overrightarrow{res}_{n''} = (0, 1, 0)$, for instance, denotes that only one resource is seized, namely a bed in recovery phase 1, when activity n'' is performed. The minimal and maximal zero time lags ($FS_{MIN} = 0$ and $FS_{MAX} = 0$) between the activities $n' - n''$ and $n'' - n'''$ in Figure 2 indicate that no time is allowed between the completion of the former and the start of the latter activity.

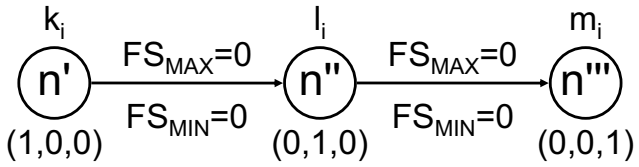

Figure 2: Representing a surgical process as a sequence of its constituent activities.

The equivalence between the RIP and the R-SCSP should now become transparent. We still have to introduce some modifications in order to complete the activity-on-the-node representation of the RIP. We have to define, for instance, a dummy start and a dummy end activity and add a $FS_{MIN} = 0$ precedence relation both between the dummy start activity and each first activity of a substituting sequence and between the last activity of such a sequence and the dummy end activity. Moreover, a $FF_{MAX} = P_{rs}^{ub} - P_{rs}^{lb} + 1$ precedence relation between the dummy start and the dummy end node needs to be specified in order to capture the project deadline. The dummy start activity is completed at time $p = P_{rs}^{lb}$. Since the surgical act inevitably needs an operating room to be performed in and the capacity of this resource is limited to 1 in the R-SCSP, we do not take the leveling of this resource into account (best equals worst). Both the peak number of beds in recovery phase 1 ($max_p\ resource_{2p} = \alpha_5$) and recovery phase 2 ($max_p\ resource_{3p} = \alpha_6$), on the contrary, have to be minimized. The procurement cost related to these resources is equal to $cost_2 = w_5/(worstvalue_5 - bestvalue_5)$ for the use of one bed in recovery phase 1 and equal to $cost_3 = w_6/(worstvalue_6 - bestvalue_6)$ for the use of one bed in recovery phase 2.

$\Rightarrow$ Assume that we have a solution to the RIP, i.e. we know for each substituted patient n

the start times $v_{n'}$, $v_{n''}$ and $v_{n'''}$, then we can construct a solution for the R-SCSP as follows: $\forall n \in N_s \Rightarrow x_{type_n,v_{n'},s} = 1$.

$\Leftarrow$ Given a solution to the R-SCSP, we can construct a solution for the RIP as follows: $\forall n \in N_s : x_{type_n,v_n,s} = 1 \Rightarrow\ v_{n'} = v_n,\ v_{n''} = v_n + k_{type_n}$ and $v_{n'''} = v_n + k_{type_n} + l_{type_n}$.□

3 Pure integer programming

Adding the IP model stated throughout Section 2 to the environment of the ILOG CPLEX 8.1 optimization library would be a first, standard approach for solving the aforementioned problem. In Section 5, we will refer to this solution method as the basic IP approach. In the remainder of this section we will introduce two variations of this standard procedure and refer to them as the preprocessed IP approach and the iterated IP approach.

3.1 Preprocessed IP

The basic IP approach could easily be enhanced on three levels: next to the modification of CPLEX-based parameters, we will exploit structure that stems from infected patients and explicitly fix variables by solving multiple knapsack problems. On average about 23% of the decision variables can be fixed to 0 during the preprocessing stage (see Section 5).

3.1.1 Parameter tuning

A first improvement involves probing and is available in the ILOG CPLEX 8.1 optimization library. Probing is a technique that looks at the logical implications of fixing each binary variable to 0 or 1. It is performed after preprocessing and before the solution of the root relaxation (ILOG 2002). Applying probing, however, can be time consuming since the probing time is somehow proportional to the difficulty of the instance. This implies that we cannot guarantee that the decrease in solution time will outperform the time needed in the probing phase. Test runs with this single enhancement, however, indicate that probing is worthwhile for the SCSP. Second, we will shift the emphasis of the IP solver towards feasibility. Since less computational effort will be spent in the proof of optimality, this feature should reduce the number of instances for which no solution could be found (see Section 5).

3.1.2 Exploit the structure inherent to the infected patients

The presence of infected patients can simplify the scheduling process of surgeries in two ways. On the one hand, there might be a possibility to merge idle periods into one large cleaning block. This implies that the number of surgeries to be scheduled is reduced and that a reduction in the number of variables is acquired. We can merge idle types for surgeon s into a new surgery type *clean* when $|B_s| - \sum_{r \in R_s} \gamma_{rs} > 0$. Recall that no additional cleaning is required when the surgeon is the last surgeon in the operating room. $|N_s|_{type_n=clean}$ is then equal to $|B_s| - \sum_{r \in R_s} \gamma_{rs}$, whereas we have to reduce the number of idle types to be scheduled for surgeon s by $|N_s|_{type_n=clean} \cdot k_{clean}$ units. On the other hand, we can limit the number of periods on which the surgery of an infected patient can start. In order to do so, we require that $|B_s| = 1$, $|N_s|_{type_n=clean} = 0$ and $|N_s|_{type_n=idle} < k_{clean}$. Note that this implies that surgeon s is the last surgeon in at least one operating room. Let i be the infected type, then we have for each operating room $r \in R_s$:

- If $\gamma_{rs} = 1$, then $\forall p: \ max\{P_{rs}^{lb}, presurgref \cdot \theta_i^{presurg}\} \leq p \leq P_{rs}^{ub} - k_i - |N_s|_{type_n=i} : \ x_{ips} = 0$.
- If $\gamma_{rs} = 0$, then $\forall p: \ max\{P_{rs}^{lb}, presurgref \cdot \theta_i^{presurg}\} \leq p \leq P_{rs}^{ub} - k_i + 1 : \ x_{ips} = 0$.

3.1.3 Identification of allowed surgery start times

Let us illustrate by means of an example that not only the allowed surgery start times of infected patients can be limited. Suppose we have to schedule the three surgeries depicted in Figure 3 (a) in the empty operating room. We can question whether the surgery of type 4 can start on period 5, as represented in Figure 3 (b). Fixing $x_{4,5,s} = 1$, however, results in dividing the operating room in two residual time sections: period 0 up to period 4 and period 9 up to 10. When we refer to knapsack A for the first time section and knapsack B for the second section, we can solve the question whether the surgery of type 4 can start on period 5 by solving a multiple knapsack problem. In this multiple knapsack problem, we are only interested in finding a feasible solution: is it possible to assign the remaining surgeries to the knapsacks so that the capacity of the knapsacks is not violated. In the illustration, we can see that it is not possible to fit a surgery of type 4 and of type 5 into the knapsacks, so that we can conclude that the decision variable $x_{4,5,s}$ must be equal to 0. Note that only the surgery duration is taken into account during the assignment process and that other constraints (e.g. instrument use) are relaxed. The

solution procedure of the multiple knapsack problem is based on a recursion function and is executed for each decision variable that is added to the model.

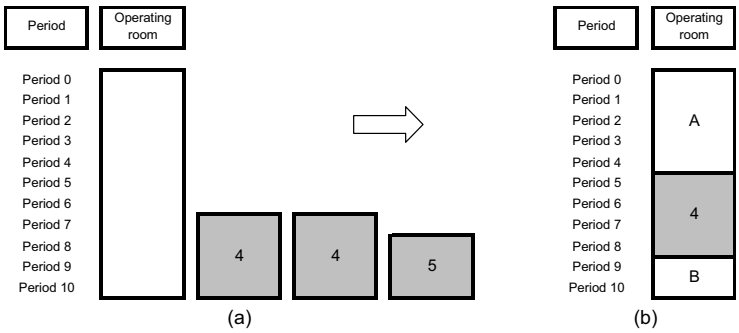

Figure 3: Identifying allowed surgery start times by solving multiple knapsack problems.

3.2 Iterated IP

In Section 5, we will indicate that the difficulty of solving the SCSP does not solely depend on the size of the patient population. In particular, the spread of the patient population over the set of doctors tends to be critical: the number of schedules resulting from sequencing 8 surgeries for one surgeon is 70 times larger than the number of schedules that should be evaluated when two surgeons have to perform 4 surgeries each. We can expect that fixing the starting period of a set of surgeries for a surgeon with a large patient population will enhance the solvability of the problem.

In the iterated IP, we will start from the preprocessed IP and continuously fix a variable set of surgeries. In particular, four decisions have to be taken. First, we have to decide on the set of surgeons for which a fixation can take place. This will depend on the number of surgeries that the specific surgeons have to perform. Second, a probability has to be determined that indicates the percentage of surgeries that will be fixed for each surgeon of the preselected set. Third, we have to specify a probability to undo the fixation of a specific surgeon. This implies that we can arrive in the setting of the preprocessed IP in which no particular surgery start times are fixed, except those that are not feasible. Finally, we have to indicate the appropriate solution time for one iteration. The heuristic algorithm tested in Section 5 fixes the start time of about 30% of the surgeries for surgeon $s : |N_s| \geq 10$. There is a 5% probability that the fixation pattern of

a surgeon is undone. Computation time for one iteration is limited to 25 seconds. A thorough testing of the influence of these solution parameters on the solution quality should be a subject for future research (see Section 6).

4 Dedicated branch-and-bound

We already mentioned the relation between the SCSP and the RCPSP in Section 2.3. Since we know that the literature provides some powerful branch-and-bound procedures for the RCPSP (e.g. Demeulemeester and Herroelen 2002, Jozefowska and Weglarz 2006), we can wonder whether implicit enumeration through branch-and-bound would also be beneficial to solve the SCSP.

4.1 A basic branch-and-bound procedure

We will structure the description of this exact but basic branch-and-bound procedure according to Agin (1966). In his generalized description of branch-and-bound algorithms, a distinction is made between the branching characteristic and the bounding characteristic. The branching characteristic guarantees that the algorithm will eventually obtain an optimal solution, whereas the bounding characteristic enables the algorithm to end up with an optimal solution without complete or explicit enumeration of all solutions. We refer to Algorithm 1 for a view of the pseudo-code.

4.1.1 Branching characteristic

We will develop a tree in which each node represents a subset of the set of solutions of the parent node. We have to decide, however, how we will partition the subset of the parent node. In particular, two decisions need to be made. First, we have to identify the operating room in which we want to schedule the next surgery. Second, we have to decide which surgery will be scheduled next. Note that this branching process results in a non-binary tree. Due to time and memory restrictions, we opt for a depth-first approach.

The choice of the operating room actually determines the shape of the surgery schedule during the optimization process. We could, for instance, decide to entirely fill up the first operating room with surgeries before we switch to a second operating room. Alternatively, we could continuously alternate between the operating rooms or choose operating rooms so that the

Algorithm 1 Basic branch-and-bound

```
list_of_types ← GET_SEQUENCE();
best_found ← 1;
r ← 1; level ← 0;
RECURSION(r, level + 1);

RECURSION(r, level);
{
index ← 0;
placed ← FALSE;
while (dominated = FALSE and elapsed_time < TILIM and index < I) do
  eligible ← FALSE;
  while (eligible = FALSE and index < I) do
    index ← index + 1;
    get surgery type using list_of_types and index;
    eligible ← CHECK_FEASIBILITY();
  end while
  if (eligible = TRUE) then
    placed ← ADD_SURGERY();
    lower_bound ← CALCULATE_LOWER_BOUND();
    if (lower_bound < best_found) then
      dominated ← DOMINANCE();
      if (dominated = FALSE) then
        if (schedule is complete) then
          best_found ← lower_bound;
          register schedule;
        else
          determine next operating room r' to schedule a surgery;
          RECURSION(r', level + 1);
        end if
      end if
    end if
  end if
  if (placed = TRUE) then
    placed ← REMOVE_SURGERY();
    if (backtracking due to domination is complete) then
      dominated ← FALSE;
    end if
  end if
end while
}
```

(intermediate) workload in each operating room is somehow leveled. Obviously, combinations of these branching schemes could be made. The scheme that incorporates alternating operating rooms will be applied during the computational testing of the algorithm (see Section 5) since we hope to detect resource conflicts near the top of the tree. The choice of the surgery type that has to be scheduled depends on the surgeon in the operating room at that period and a sequenced list of surgery types.

4.1.2 Bounding characteristic

Several bounding techniques will be applied in order to limit the tree search. Next to the introduction of a lower bound calculation and a fathoming rule, we will also check the viability of

dominance rules.

It is not straightforward to calculate a tight lower bound due to the multiple objectives. In fact, we will calculate a lower bound for each objective and merge them into one value. For each objective $j \in J$ we will calculate the corresponding α_j of the partial schedule. Next, we will try to augment α_j for $j \in J : j \leq 4$ by analyzing the set of surgeries that still have to be scheduled. In particular, we will try to add the remaining surgeries to the schedule in such a way that the corresponding α_j is kept as small as possible. This will be done for each objective individually and with relaxation of the instrument, infection and bed constraints in order to speed up computation time. Since augmenting α_j for $j \in J : j \geq 5$ can still be computationally expensive and the lower bound has to be calculated frequently, we do not consider the augmenting step for these objectives. When the value of α_j is smaller than $bestvalue_j$ (due to the relaxations), we will set $\alpha_j = bestvalue_j$.

A second bounding procedure is captured in a fathoming rule. This fathoming rule in fact incorporates the logic described in Section 3.1.2. When an infected patient or an idle period is scheduled, we will investigate whether we still can obtain a feasible schedule or not. This rule implies that we will determine the minimal number of idle periods needed to generate a feasible schedule. If this number is larger than the remaining number of idle periods, we will fathom the node and consequently backtrack.

Finally, we also thought of two dominance rules. In a first dominance rule, we adapted the *cutset dominance rule* that was introduced by Demeulemeester and Herroelen (1992) to the SCSP setting. Preliminary computational results, however, revealed that this dominance rule does not significantly increase the performance of the branch-and-bound algorithm. A reason can be found in the fact that the domination of a partial schedule only results in limited backtracking. Moreover, the domination typically occurs when the tree already has a substantial depth, which diminishes the cutting power. The application of the dominance rule furthermore suffers from the extensive amount of data that continuously has to be stored and retrieved. Due to the many conditions that have to be satisfied, this dominance rule also occurs with a limited frequency in comparison with the computational effort needed to check the conditions. Based on these considerations, we will not take the cutset-based dominance rule into account during the evaluation of the test set. In the second dominance rule, in which we will try to tackle symmetry, we do not need to register partial schedules during the tree exploration. Instead,

we will only focus on the current partial schedule and try to identify for a specific surgeon s two surgeries for which a swap is favorable. Below we will show that the favorability of a swap does not necessarily imply that there is a decrease in the objective function. When we assume that $|J| = 6$ and patient $n' : type_{n'} = i'$ is the patient that was just added to the schedule, we want to find a patient $n : type_n = i$ for which $(v_n < v_{n'})$ and $i \neq i'$, $k_i = k_{i'}$, $l_i = l_{i'}$ and $m_i = m_{i'}$. Furthermore, instrument requirements should be exactly the same for i and i'. When two surgeries can be identified that satisfy these conditions, we will check the feasibility of the swap with respect to pre-surgical tests and infections when needed (i or $i' \in I_{presurg}$ / i or $i' \in I_{bact}$). When feasibility conflicts do not occur, we can calculate the favorability of the swap:

- $\triangle$ *objective* $1 = \frac{w_1 \cdot [(\theta_{i'}^{child} - \theta_i^{child}) \cdot (v_{n'} - v_n)]}{worstvalue_1 - bestvalue_1}$
- $\triangle$ *objective* $2 = \frac{w_2 \cdot [(\theta_{i'}^{prior} - \theta_i^{prior}) \cdot (v_{n'} - v_n)]}{worstvalue_2 - bestvalue_2}$
- $\triangle$ *objective* $3 = \begin{cases} 0, \text{ if } (travelref \leq v_n) \text{ or } (travelref > v_{n'}) \\ w_3 \cdot (\theta_i^{travel} - \theta_{i'}^{travel})/(worstvalue_3 - bestvalue_3), \text{ otherwise} \end{cases}$

Since both $l_i = l_{i'}$ and $m_i = m_{i'}$, a swap has no effect on objective 4, 5 or 6. We can now state the dominance rule in Theorem 2. The proof of this theorem is trivial and will hence be omitted.

Theorem 2. *When $\triangle$ objective 1 + $\triangle$ objective 2 + $\triangle$ objective 3 > 0, the current partial schedule is dominated and the algorithm can backtrack until $x_{iv_n s} = 0$. When $\triangle$ objective 1 + $\triangle$ objective 2 + $\triangle$ objective 3 $= 0$ and $i > i'$ (tie break), the current partial schedule is dominated and the algorithm can backtrack until $x_{iv_n s} = 0$.*

4.2 Nested branch-and-bound

In the nested branch-and-bound procedure, we will adjust the basic enumeration algorithm on two levels. On the one hand, we will introduce a heuristic in order to rapidly improve the surgery schedule. This way, we can focus the computational effort on the pool of qualitative schedules. On the other hand, we will try to increase the ability to find at least one feasible solution within the time limit. We want to stress that the modifications do not damage the exact nature of the algorithm.

4.2.1 Heuristic to overcome immobility

Due to the lack of strong bounding characteristics, the basic branch-and-bound procedure does not easily succeed in backtracking to the top of the solution tree. This implies that the first scheduled surgeries are somehow immobile and can limit the quality of the schedules that will be explored within the time range. For this reason, we developed a heuristic that should tackle this immobility problem and is used during the tree generation. Since this heuristic will be a branch-and-bound algorithm too, we will have a nested algorithm.

The heuristic will be applied each time a new best solution is encountered and the corresponding schedule is registered. In particular, we will fix the sequence of surgeries in a certain amount of operating rooms, whereas we will restart the scheduling process in the other operating rooms. When we take a look at Figure 4, we can see that the sequence of surgeries for operating rooms 1, 2, 4 and 5 is fixed and that we have to reschedule surgeries for operating rooms 3 and 6. A second branch-and-bound algorithm will be used to enumerate the feasible sequences within the emptied operating rooms. When the searching process of this second algorithm ends, the first branch-and-bound procedure will continue the exploration of the tree from the leaf where the heuristic was evoked.

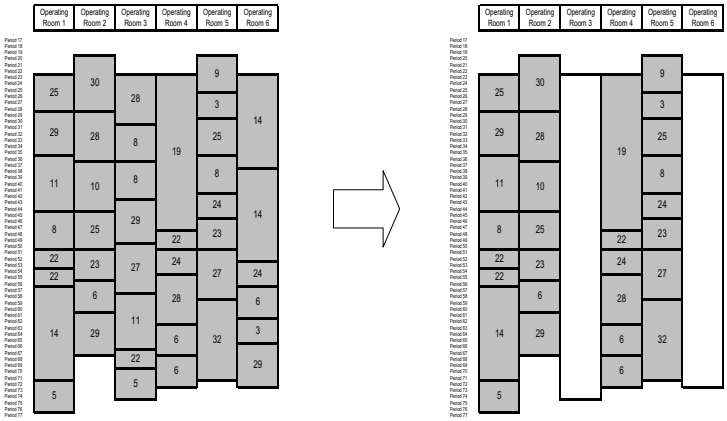

Figure 4: Illustrating the fixation of a surgery schedule.

The problem in Figure 4 is reduced to finding feasible schedules for a SCSP with two operating rooms. This reduction in problem size obviously empowers the capabilities of the second branch-and-bound algorithm. Moreover, we can take the capacity of the fixed operating rooms into account so that the lower bound calculation will sooner collide with the currently best solu-

tion found and constraints will be frequently violated. However, when the number of surgeries that has to be rescheduled is still large, the problem of immobility can still occur in the second branch-and-bound tree. Hence, we will continuously truncate the exact procedure, shuffle the list of surgery types that have to be scheduled and restart the second branch-and-bound algorithm. This way, other surgeries will be scheduled near the top of the tree and the immobility disappears.

We still have to decide on which operating rooms we will use for rescheduling. In order to retain the limited problem size, we will allow for at most 3 operating rooms to be rescheduled. With respect to Figure 4, small runs of the second branch-and-bound procedure will be executed for each combination of 3 operating rooms (1,2,3 - 1,2,4 - 1,2,5 - 1,2,6 up to 4,5,6), for each combination of 2 operating rooms (1,2 - 1,3 - 1,4 up to 5,6) and for each single operating room. Whenever a new best solution is encountered, the entire procedure is repeated.

4.2.2 Finding an applicable start sequence

An obvious approach to reduce the number of instances for which no feasible solution could be found within the time limits would be to continuously restart the branch-and-bound algorithm with a different sequence in the list of surgery types and fix that list when the first feasible schedule is encountered. Unfortunately, preliminary results indicated that there was still a significant amount of instances for which this approach was insufficient. The main reason is that the recovery capacity constraints for those instances seem to be very tight: there are only few surgery schedules for which these constraints are not violated. The modification we will introduce is based on loosening the recovery constraints, i.e. we will increase the capacity in recovery phase 1 ($= cap_l$) and recovery phase 2 ($= cap_m$), and run the branch-and-bound procedure until a complete schedule is generated. At this point, two possible situations may occur. First, it is possible that the schedule is feasible with respect to the original or tight recovery capacity constraints. In this case, there is no problem and we could definitively tighten the constraints again and proceed the branch-and-bound algorithm. Second, we could end up with a schedule that is only feasible with respect to the loosened constraints so that we should wonder whether we could easily generate a similar schedule that does not violate the original constraints. At this point, the heuristic comes into play once again. The removal of certain surgeries from the schedule will result in a partial schedule that satisfies the tight constraints.

We will use the second branch-and-bound procedure to reschedule the removed surgeries in such a way that the original constraints are not violated. When we are able to find such a tight schedule, we can definitively tighten the constraints again. Otherwise, we should continue the tree generation with loosened constraints and repeat the heuristic procedure when the next (possibly infeasible) complete schedule is encountered.

We have to be careful to what extent we will loosen the recovery constraints. On the one hand, it might be possible that the increase in capacity is still not sufficient for the algorithm to easily find (probably infeasible) complete schedules when the sequence of the surgery list is poor. On the other hand, when too much capacity is added, the algorithm will probably generate schedules that are so bad that the immobility heuristic will not be able to find a schedule that does not violate the tight constraints, since at most 3 operating rooms will be emptied. In order to overcome both difficulties, we will increase the capacity of each recovery phase with only one bed and restart, at regular time points, the entire branch-and-bound procedure with a randomly shuffled list of surgery types. It should be clear that we will only restart the algorithm when no feasible schedule (w.r.t. the tight constraints) is encountered yet. The shuffle is random since we still have no idea how a good list looks like.

4.3 Iterated branch-and-bound

In the previous section, the branch-and-bound algorithm was only restarted when a feasible solution was not yet found. We could argue whether it would be advantageous to restart the algorithm from time to time, even when a feasible solution was already encountered. This implies that we will shift from an exact branch-and-bound procedure towards an iterated and hence truncated branch-and-bound algorithm.

The heuristic continuously truncates a branch-and-bound procedure and restarts a new branch-and-bound procedure. The settings of the new tree generation algorithm can differ from the previous procedure on three levels. First, there might be a difference in the branching scheme (see 4.1.1). Second, it is possible that the surgery scheduling process starts in a different operating room. Finally, there might be, and probably will be, a change in the sequence of the list of surgery types. This change will be triggered by a shuffle function and the sequence of surgery types that corresponds to the schedule with the best solution yet found. This shuffle function will perform a number of swaps in the sequence of surgery types, based on a multi-

nomial probability distribution that allows for incremental changes. We actually argue that a better schedule can be found by minor changes in the provisional best surgery list. This way, we should encourage progression towards a local (but hopefully also global) optimum. In order to get out of local optima, we reset the provisional best solution value after 25000 trials, perform a large number of random swaps on the surgery list and restart the entire procedure. Obviously, we register the overall best solution encountered during the entire algorithmic search.

Truncation of the tree generation algorithm will be done by limiting its ability to backtrack. This ability is expressed by the number of surgery removals during an iteration. We will randomly distinguish between a small, medium or large number of allowed removals. The choice of this number will be renewed each time the number of trials without improvement in the objective function is larger than a limit. This is a self-regulating feature since we do not know whether it is advantageous to have short or extensive tree explorations. The idea is to maintain the backtrack ability of the algorithm as long as new solutions are encountered frequently. Whenever the removal limit is exceeded, the algorithm backtracks to level 0 and a new branch-and-bound iteration is initiated.

In order to reduce the number of instances for which no solution can be found in regular time, tight constraints will be loosened until a feasible schedule is encountered that satisfies the original constraints. The approach is similar to that of the nested branch-and-bound procedure (see Section 4.2.2). When the number of trials exceeds the corresponding limit and the entire algorithm is restarted, we will not loosen the constraints when a feasible schedule was already encountered in one of the previous iterations. The relevance of the immobility heuristic in the iterated branch-and-bound procedure is twofold. On the one hand, it will be called in the search for the first feasible schedule. On the other hand, we will execute the heuristic whenever the best solution value is reset, i.e. after 25000 trials. Note that the schedule that will be used in the heuristic does not correspond to the overall best solution value but to the solution value that was reset. This way we encourage the application of the heuristic on diversified schedules.

Several features could be turned off for a specified time interval. This implies that the allowed number of removals possibly doesn't change, that the immobility heuristic is only used in the search for a feasible solution or that the value of the provisional best solution is not reset. A study of the parametrical influence on the solution quality should be a study for future research (see Section 6).

5 Computational Results

5.1 Instance Generation

The instances that we will generate in order to test the solution procedures are based on data from the surgical day-care center of the university hospital Gasthuisberg, situated in Leuven, Belgium. This medical facility has already been the subject of research in a case study of Beliën, Demeulemeester and Cardoen (2006) in which the master surgery schedule is visualized, based on data of 2004. The day-care center opens at 7 a.m. and closes at 7 p.m. The operating theater consists of 8 operating rooms, opened between 8 a.m. and 5 p.m. Furthermore, there are 8 beds in recovery phase 1 and 12 beds in recovery phase 2.

We will use patient-related data gathered in 2005 and make a distinction between 17 of the most important medical disciplines or entities (e.g. orthopaedics, gynaecology, dermatology,...). This implies that our calculations incorporate over 25000 surgeries and about 15000 hours of total net operating time. For each medical discipline and their surgery types we can calculate the probability of occurrence. Furthermore, for each surgery type, the planned surgical duration (including anaesthesia, skin-to-skin time, after care and cleaning), the planned time in recovery phase 1 and phase 2, the required bottleneck instruments and the corresponding sterilization time is known. All time-related data are expressed in five-minute periods. Contrary to the recovery length, the duration of a surgery is at least equal to one period. Probabilities concerning children, priority, travel distance, pre-surgical tests and infections, however, are only occasionally registered up to now and hence suggested by the health manager, based on his experience.

The test instances will be generated according to one of eight possible patterns. These patterns are represented in Figure 5 and can differ on 3 levels. On the first level, we will distinguish in the assignment of surgeries over the operating theater. On the one hand, we will entirely fill up an operating room before we go on with the next operating room (=depth). This implies that we will have only few operating rooms in which surgeries will be scheduled. However, when an operating room is used, it will probably be used until closing time. Note that we will limit $|N_s|$ to 15 patients. On the other hand, we will head for a simultaneous use of the entire operating theater (=width). This implies that all operating rooms will be in use although there will probably not be enough surgeries to fill them up entirely. For the real-sized instances, this distinction will somehow become superfluous since the entire operating theater

will be (nearly) fully booked. The second level is used to distinguish between master surgery schedules with frequent switches of surgeons in the operating rooms and surgery schedules in which a surgeon switch is rare. This can be important since not only the number of surgeries to be scheduled influences solvability, but also the number of surgeries for each surgeon. Observe, for instance, that sequencing 8 surgeries of one surgeon in one operating room can lead to $8! = 40320$ schedules, whereas sequencing an operating room with 8 surgeries exactly divided over 2 surgeons results in only $4! \cdot 4! = 576$ sequences. Note that this reasoning also holds for the decision on level 1: few operating rooms with many surgeries will result in more schedules than when the surgeries are divided over the total set of operating rooms. Finally, on the third level, we will determine whether the objectives of the instance will be equally weighted or not. The sum of the weights in the instances will be equal to 1 (see Section 2.1.2).

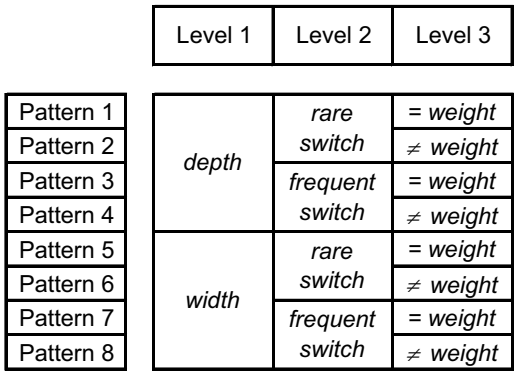

Figure 5: Representation of the patterns for instance generation.

We generated a test set that consists of 8 patterns · 2 instances · 14 sizes = 224 test instances. The size of an instance indicates the number of surgeries that has to be scheduled. The sizes range from 20, 25, 30,... up to 85 surgeries. This implies that the instances vary from rather small to real-life problems. The algorithm is written in MS Visual C++.NET and is linked with the ILOG CPLEX 8.1 optimization library in order to get the best and worst values of the individual objectives. We will limit the allowed computation time for the solution procedures to 300 seconds per instance on a 2,8 GHz Pentium 4 PC with the Windows XP operating system. Solution gaps will be calculated with respect to the optimal solution. However, for 4.9% of the instances no optimal solution could be found within 5 hours of computation time (=18000 seconds) using the preprocessed IP procedure. Instead, we will take the value of the current best node in the tree

that still has to be explored as a lower bound for the optimal value. Note that the relaxation of the problem could result in a negative value since α_j could be smaller than the best (integer) value of objective $j \in J$. In other words, $w_j \cdot ((\alpha_j - bestvalue_j)/(worstvalue_j - bestvalue_j))$ could be negative. The relaxed solution value of an instance however will never be smaller than $-\sum_j w_j.(bestvalue_j/(worstvalue_j - bestvalue_j))$.

We can wonder whether the structure of an instance significantly determines the solvability or whether this solvability depends on random and uncontrolled factors. We will investigate this issue using the basic integer programming procedure. From Figure 6, we clearly see that the number of surgeries that has to be scheduled negatively influences the probability to find an optimal solution. However, as can be seen in Figure 7 (a) and (b), not only the number of surgeries affects solvability. There seems to be a major fluctuation based on the patterns.

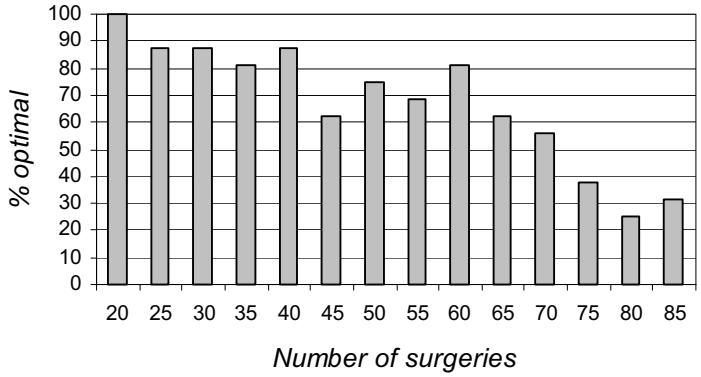

Figure 6: Influence of number of surgeries on solvability.

When we take a closer look at Figure 7 (b), we see that instances that are generated in depth (level 1) are harder to solve. We already mentioned the reason for this phenomenon: sequencing 10 surgeries in one operating room results in many more possible schedules than sequencing 5 operating rooms with each 2 surgeries. Furthermore, we notice that the presence of surgeon switches in an operating room facilitates the search to optimality. An analogous explanation applies in this case. Finally, instances characterized by unequal weights tend to be slightly easier to solve than those with equal weights (level 3). The diversification of the weights may enable the solution procedure to exhibit features of a sub-optimal lexicographic methodology. This means, in the extreme, that the problem is optimized for the most important (largest weight) objective first and that the solution space is limited to schedules which set this objective to

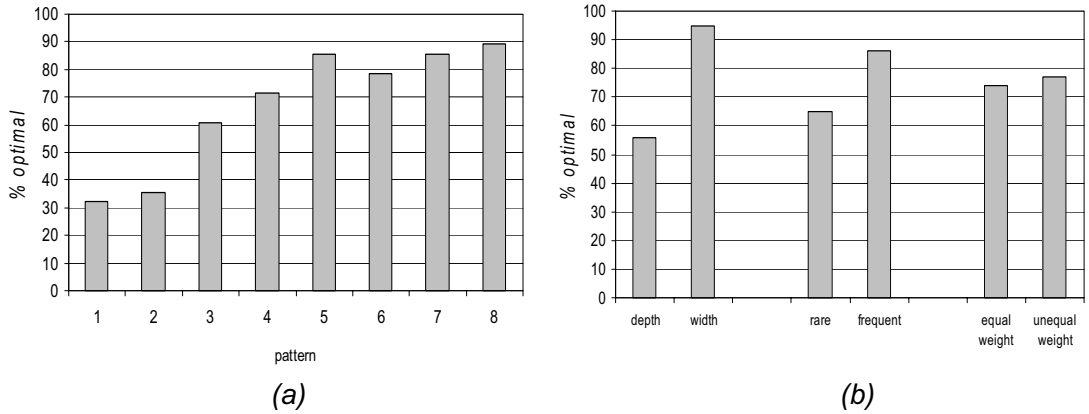

Figure 7: Influence of patterns on solvability.

its best value. Then the next most important objective is optimized and the solution space is consecutively limited. When weights take proportions so that trade-offs are unlikely to happen, we somehow create an equivalent situation which is easier to solve since the number of schedules to be evaluated is limited.

5.2 Comparison of solution approaches

Results of the computational experiment are summarized in Table 1. We will, though, try to visualize the most important findings in order to improve comprehensibility.

When we look at the percentage of instances solved to optimality in Figure 8, we clearly see that the preprocessed IP outperforms the other solution approaches. However, since both the iterated IP and the iterated branch-and-bound are heuristics and hence inherently unable to prove the optimality of solutions, we should also take a look at the number of instances for which there is no gap between the solution value and the optimum. Since the solutions that are proven to be optimal inherently have such a zero gap, this percentage should be at least the equivalent of the percentage of instances solved to optimality. Concerning this zero solution gap criterion, we may conclude that the preprocessed IP is equivalent to the iterated IP and outperforms the basic IP procedure, whereas the iterated branch-and-bound is the best of the implicit enumeration algorithms. The preprocessed IP and iterated IP, however, outperform the iterated branch-and-bound.

The percentage of instances for which no solution could be found within the time limits is

Table 1: Computational results.

		solution	time	abs gap solution	abs gap relaxation
Basic IP	average	0.134	119.586	0.043	0.056
67% opt - 4% no sol	median	0.083	35.062	0.000	0.042
71% zero sol gap	st. dev.	0.190	134.305	0.166	0.051
Preprocessed IP	average	0.106	95.883	0.016	0.043
74% opt - 1% no sol	median	0.080	6.938	0.000	0.035
82% zero sol gap	st. dev.	0.121	129.848	0.089	0.039
Iterated IP	average	0.097	/	0.007	/
0% no sol	median	0.078	/	0.000	/
81% zero sol gap	st. dev.	0.084	/	0.024	/
Basic B&B	average	0.289	208.042	0.198	/
33% opt - 16% no sol	median	0.161	300.000	0.071	/
35% zero sol gap	st. dev.	0.335	134.871	0.299	/
Nested B&B	average	0.130	208.394	0.039	/
33% opt - 0% no sol	median	0.099	300.000	0.009	/
41% zero sol gap	st. dev.	0.119	133.616	0.063	/
Iterated B&B	average	0.115	/	0.024	/
0% no sol	median	0.085	/	0.000	/
58% zero sol gap	st. dev.	0.119	/	0.065	/

also depicted in Figure 8. The iterated IP, nested branch-and-bound and the iterated branch-and-bound seem to be the only procedures that guarantee the occurrence of at least one feasible schedule. It should be clear that lacking a feasible solution can be a serious drawback for the use of an algorithm in practice.

We can question whether the difference in solution quality is rightfully represented by the percentage of instances for which the solution value is equal to the optimal value. Solutions that are close to the optimum can still be relevant and consequently influence the overall solution quality of a methodology. We will capture this solution quality by the average gap between the solution values and the optimal values and the standard deviation of these gaps (Figure 9).

From Figure 9, we can see that the iterated IP outperforms all other solution approaches, both on the level of the average absolute solution gap and the corresponding standard deviation. This implies that these suboptimal solutions will probably be close to the optimal value.

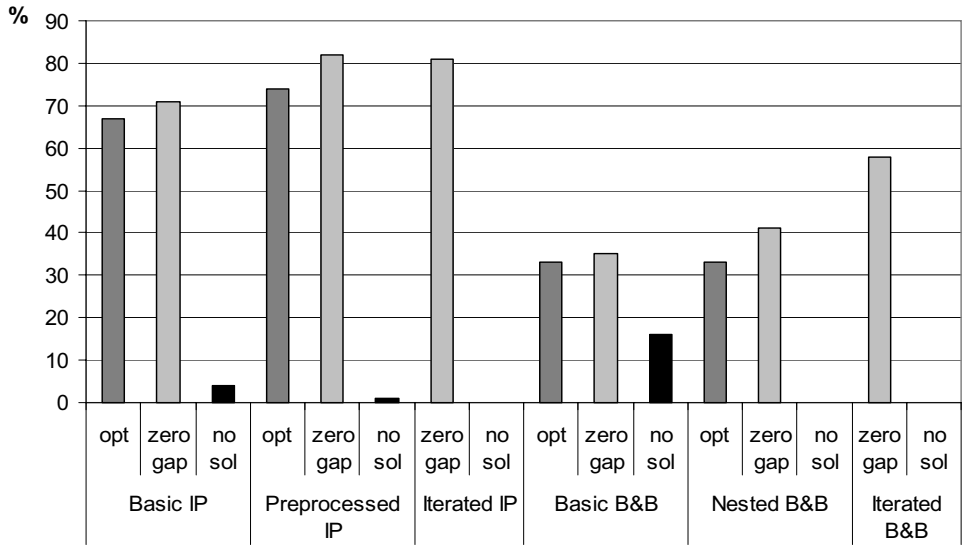

Figure 8: Visualizing performance criteria of the solution approaches.

In contrast with the iterated IP, Figure 9 indicates that the basic branch-and-bound procedure is insufficient to provide qualitative solutions and hence illustrates the difficulty in solving the SCSP. Note that the enhancements introduced to the nested and iterated branch-and-bound procedures result in algorithms that have a reasonable performance on both criteria. Both algorithms perform better than the basic IP. A reason why their standard deviation is smaller than those of the basic IP and the preprocessed IP can be found in the reduction of instances for which no solution could be obtained.

We can summarize the computational results by stating that the iterated IP outperforms the basic and preprocessed IP procedure and that the iterated branch-and-bound algorithm outperforms the other branch-and-bound procedures. When priority is fixed on proving optimality, the preprocessed IP model should be chosen. However, when the proof of optimality is of minor importance and stability and solution quality are of major importance to the decision maker, the application of the iterated IP should be considered.

6 Conclusions and Future Research

In this research paper a surgical case scheduling problem is formulated that originated from the suggestions and needs of the surgical day-care center at the university hospital Gasthuisberg in

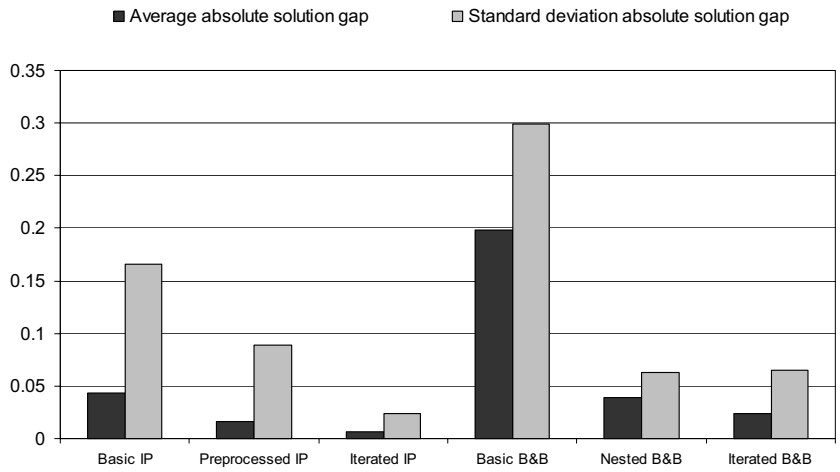

Figure 9: Visualizing the average absolute solution gap and the corresponding standard deviation.

Leuven (Belgium). Multiple, possibly contradictory objectives are accurately transformed into a multi-objective function and constraints are added to the model. We have shown that this surgical case scheduling problem is NP-hard and hence hard to solve. Two solution methodologies were investigated in order to minimize the multiple objective function. On the one hand, integer programming models were formulated in which the logical structure of the problem was exploited. On the other hand, dedicated algorithms were developed using a branch-and-bound methodology. It turned out that two effective modifications drastically upgraded the poor performance of the basic branch-and-bound algorithm. All solution approaches were computationally evaluated and compared to each other. Especially the iterated IP procedure proved to provide promising results. In the future, the influence of the parametrical settings of the algorithms on the solution quality should be examined in more detail.

Instead of scheduling individual surgeries, we could also try to schedule predetermined groups of surgeries. Such a group can, for instance, be seen as the entire, somehow sequenced population of patients that has to be scheduled for a specific surgeon. When we call such a group a column, it is not hard to make the link with column generation. Formulating a problem using columns, i.e. a huge number of variables, and solving it by column generation tends to result in a tightened LP relaxation (Barnhart et al. 1998). When the resulting relaxation is tight, the probability to find the optimal integer solution in a reasonable amount of time increases. In order to test this proposition, we want to develop and implement a column generation algorithm

for the SCSP. However, since column generation does not guarantee integrality of the variables, we will also have to combine this algorithm with a branching scheme. This methodology is referred to as branch-and-price and constitutes the main topic for future research.

Recall that the sequencing step of the surgical case scheduling problem is preceded by an assignment step in which patients are assigned to surgery days. The development of online assignment policies will definitively be investigated in the future. However, until the branch-and-price algorithm is developed, the focus remains on the sequencing phase.

One could argue that in a hospital scheduling environment planning often substantially deviates from reality. It is, for instance, not possible to know the exact duration of a patient's surgery, so that estimations are needed. Interesting findings in order to deal with such surgery duration variability can, for instance, be found in Hans et al. (2005). In the SCSP, however, surgery durations are deterministic. Since we are investigating a day-care environment, in which difficult, rare and highly uncertain surgeries are typically not performed and procedures are more or less standardized, the deterministic approach should not present a major drawback.

Acknowledgements

We are grateful to Pierre Luysmans of the surgical day-care center at Gasthuisberg for introducing this practical scheduling problem and providing the data in order to build a structured test set. His suggestions and experience contributed to the practical value of this research. We acknowledge the support given to this project by the Bijzonder Onderzoeksfonds of the Katholieke Universiteit Leuven.

Appendix

In this Appendix we will state the symbols used throughout the mathematical formulations of the paper. We will distinguish between indices, sets, decision variables, help variables and functions, and data parameters.

Indices:

i, i', i":	surgery type	*e*:	instrument type	*r*:	operating room
n, n', n":	patient	*j*:	objective	*b*:	infection
p, p':	period	*o*:	resource	*s*:	surgeon

Sets:

I:	set of surgery types
I_s:	set of surgery types for surgeon s
I_r:	set of surgery types that can be performed in operating room r
I_e:	set of surgery types for which instrument e is needed
I_{child}:	set of surgery types for children
I_{prior}:	set of surgery types for prioritized patients
I_{travel}:	set of surgery types for travel patients
I_{bact}:	set of surgery types with bacterial infection ($b \neq 0$)
$I_{presurg}$:	set of surgery types with pre-surgical tests
R:	set of operating rooms
R_s:	set of operating rooms for surgeon s
S:	set of surgeons
E:	set of instrument types
J:	set of objectives: $\forall j \in J : worstvalue_j \neq bestvalue_j$
N:	set of patients to be scheduled
N_s:	set of patients to be scheduled for surgeon s
B_s:	set of infections represented in patient population of surgeon s

Decision variables:

$$x_{ips} = \begin{cases} 1 & \text{if a surgery of type } i \text{ starts on period } p \text{ by surgeon } s \\ 0 & \text{otherwise} \end{cases}$$

Help variables and functions:

α_1:	$\sum_{n \in N} \Theta_{type_n}^{child} \cdot \upsilon_n$ (=periods)
α_2:	$\sum_{n \in N} \Theta_{type_n}^{prior} \cdot \upsilon_n$ (=periods)
α_3:	$\sum_{n \in N: \upsilon_n < travelref} \Theta_{type_n}^{travel}$ (=patients)
α_4:	$\sum_{n \in N} max[0, \upsilon_n + k_{type_n} + l_{type_n} + m_{type_n} - 1 - P^{ub}]$ (=periods)
α_5:	peak number of beds used in recovery phase 1 (=beds)
α_6:	peak number of beds used in recovery phase 2 (=beds)
υ_n:	starting period of surgery of patient n
RFI_j:	room for improvement of objective j
$\|Set\|$:	number of elements in Set
$\|Set\|_{condition}$:	number of elements in Set for which *condition* is true
$resource_{op}$:	units of resource o needed in period p

Data parameters:

$$\tau_{irs} = \begin{cases} 1 & \text{if a surgery of type } i \text{ is infected and } s \text{ is not the last surgeon in operating room } r \\ 0 & \text{otherwise} \end{cases}$$

$$\gamma_{rs} = \begin{cases} 1 & \text{if surgeon } s \text{ is the last surgeon in operating room } r \\ 0 & \text{otherwise} \end{cases}$$

$$\theta_i^{child} = \begin{cases} 1 & \text{if } i \in I_{child} \\ 0 & \text{otherwise} \end{cases} \qquad \theta_i^{prior} = \begin{cases} 1 & \text{if } i \in I_{prior} \\ 0 & \text{otherwise} \end{cases}$$

$$\theta_i^{travel} = \begin{cases} 1 & \text{if } i \in I_{travel} \\ 0 & \text{otherwise} \end{cases} \qquad \theta_i^{presurg} = \begin{cases} 1 & \text{if } i \in I_{presurg} \\ 0 & \text{otherwise} \end{cases}$$

P^{lb}:	opening period of the day-care center
P^{ub}:	closing period of the day-care center
P_{rs}^{lb}:	starting period for surgeon s in operating room r
P_{rs}^{ub}:	closing period for surgeon s in operating room r
k_i:	length of surgery of type i (periods)
l_i:	length of recovery phase 1 for surgery type i (periods)
m_i:	length of recovery phase 2 for surgery type i (periods)
$clean$:	cleaning type (= merger of k_{clean} idle types)
$presurgref$:	reference period for pre-surgical tests
$travelref$:	reference period for travel patients
$ster_e$:	periods needed to sterilize instrument of type e
cap_l:	capacity of recovery phase 1 (patients)
cap_m:	capacity of recovery phase 2 (patients)
cap_e:	number of instruments of type e available
$nrtravelpat$:	number of travel patients in the population
$idle$:	idle type (k_{idle}=1 period)
mrp:	latest period on which any patient can possibly be in recovery
$bestvalue_j$:	best possible value for α_j
$worstvalue_j$:	worst possible value for α_j
$type_n$:	surgery type of patient n
w_j:	weight of objective j
$cost_o$:	procurement cost per unit of resource o
$bact_i$:	infection for surgery type i, if $bact_i = 0$: no infection occurs

References

Agin, N. 1966. Optimum seeking with branch and bound, *Management Science* **13**: 176–185.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P. and Vance, P. H. 1998. Branch-and-price: Column generation for solving huge integer programs, *Operations Research* **46**: 316–329.

Beliën, J. and Demeulemeester, E. 2006. A branch-and-price approach for integrating nurse and surgery scheduling. To appear in *European Journal of Operational Research*.

Beliën, J. and Demeulemeester, E. 2007. Building cyclic master surgery schedules with leveled resulting bed occupancy, *European Journal of Operational Research* **176**: 1185–1204.

Beliën, J., Demeulemeester, E. and Cardoen, B. 2006. Visualizing the demand for various resources as a function of the master surgery schedule: A case study, *Journal of Medical Systems* **30**: 343–350.

Blake, J. T. and Carter, M. W. 2002. A goal programming approach to strategic resource allocation in acute care hospitals, *European Journal of Operational Research* **140**: 541–561.

Blake, J. T., Dexter, F. and Donald, J. 2002. Operating room manager's use of integer programming for assigning block time to surgical groups: A case study, *Anesthesia and Analgesia* **94**: 143–148.

Blake, J. T. and Donald, J. 2002. Mount Sinai hospital uses integer programming to allocate operating room time, *Interfaces* **32**: 63–73.

Carter, M. 2002. Health care: Mismanagement of resources, *ORMS Today* **19**: 26–32.

Demeulemeester, E. and Herroelen, W. S. 1992. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem, *Management Science* **38**: 1803–1818.

Demeulemeester, E. and Herroelen, W. S. 2002. *Project scheduling - A research handbook*, Kluwer Academic Publishers, Boston, MA.

Garey, M. R. and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman & Co., San Francisco, CA.

Guinet, A. and Chaabane, S. 2003. Operating theatre planning, *International Journal of Production Economics* **85**: 69–81.

Hans, E. W., Wullink, G., van Houdenhoven, M. and Kazemier, G. 2005. Robust surgery loading, *Technical Report Beta-wp141*, Department of Operational Methods for Production and Logistics, University of Twente.

Harris, R. A. 1985. Hospital bed requirements planning, *European Journal of Operational Research* **25**: 121–136.

Hsu, V., de Matta, R. and Lee, C.-Y. 2003. Scheduling patients in an ambulatory surgical center, *Naval Research Logistics* **50**: 218–238.

ILOG 2002. *ILOG CPLEX 8.1 User's Manual*.

Jebali, A., Alouane, A. B. H. and Ladet, P. 2006. Operating rooms scheduling, *International Journal of Production Economics* **99**: 52–62.

Jozefowska, J. and Weglarz, J. 2006. *Perspectives in modern project scheduling*, Springer, New York.

Kim, S.-C. and Horowitz, I. 2002. Scheduling hospital services: The efficacy of elective-surgery quotas, *Omega - The International Journal of Management Science* **30**: 335–346.

Marcon, E., Kharraja, S. and Simonnet, G. 2003. The operating theatre planning by the follow-up of the risk of no realization, *International Journal of Production Economics* **85**: 83–90.

Neumann, K., Schwindt, C. and Zimmerman, J. 2003. *Project Scheduling with Time Windows and Scarce Resources*, Springer, New York.

Ozkarahan, I. 2000. Allocation of surgeries to operating rooms using goal programming, *Journal of Medical Systems* **24**(6): 339–378.

Sainfort, F., Blake, J., Gupta, D. and Rardin, R. L. 2005. Operations research for health care delivery systems. WTEC Panel Report.

Sier, D., Tobin, P. and McGurk, C. 1997. Scheduling surgical procedures, *Journal of the Operational Research Society* **48**: 884–891.

Weiss, E. N. 1990. Models for determining estimated start times and case orderings in hospital operating rooms, *IIE Transactions* **22**: 143–150.