Minimizing the Waiting Time for Emergency Surgery

J.T. van Essen[a,*], E.W. Hans[a], J.L. Hurink[a], A. Oversberg[b]

[a] *Center for Healthcare Operations Improvement and Research (CHOIR), University of Twente, Enschede, The Netherlands*
[b] *University of Cologne, Germany*

Abstract

Hospitals aim to deliver the highest quality of care. One key priority is to schedule emergency surgeries as quickly as possible, because postponing them generally increases a patient's risk of complications and even death. In this paper, we consider the case that emergency surgeries are scheduled in one of the elective Operating Rooms (ORs). In this situation, emergency patients are operated once an ongoing elective surgery has finished. We denote these completion times of the elective surgeries by 'break-in-moments' (BIMs). The waiting time for emergency surgeries can be reduced by spreading these BIMs as evenly as possible over the day. This can be achieved by sequencing the surgeries in their assigned OR, such that the maximum interval between two consecutive BIMs is minimized. In this paper, we discuss several exact and heuristic solution methods for this new type of scheduling problem. However, in practice, emergency surgeries arising throughout the day and the uncertainty of the durations of elective surgeries, may disrupt the initial schedule. As a result, the completion times of the elective surgeries, and therefore, the BIMs change, leading also to a change of the maximum distance between two BIMs. To estimate this effect and investigate the robustness of the created schedules, we conduct a simulation study. Computational results show that the best approaches reduce the waiting time of emergency surgeries by approximately 10%.

Keywords: Operating Rooms, Emergency Surgeries, Job Shop Scheduling, Heuristics

1. Introduction

The Operating Room (OR) is one of the most expensive resources of any hospital. Surgical costs typically account for approximately 40% of the total hospital resource costs [1], while surgeries typically generate around 67% of hospital revenues [2]. Approximately 60% of all patients visit the OR department at some point during their stay [3]. Therefore, it is important to control the surgery costs while maintaining a high quality of care. Managing the OR department is challenging due to conflicting priorities and preferences of stakeholders, and the use of scarce resources. Additionally, the demand for surgical services is increasing due to the ageing population. Therefore, we need planning and scheduling methods to increase the efficiency of OR departments; see Cardoen et al. [4] and Hulshof et al. [5] for an overview on OR planning and scheduling.

To maintain a high quality of care it is important to schedule emergency surgeries as quickly as possible. Most hospitals deal with emergency surgeries by reserving some OR-capacity. This can be done in three ways: (1) dedicating an entire OR to emergency surgeries, (2) scheduling the emergency surgeries in one of the elective ORs or (3) a combination of (1) and (2). In situation (1), arriving emergency patients are operated immediately if the emergency OR is empty, but if the emergency OR is occupied, they have to wait until the ongoing surgery has finished. In situation (2), arriving emergency patients can be operated on once an ongoing elective surgery has finished. We call these completion times of the elective surgeries 'break-in-moments' (BIMs). Situation (3) is a combination of situations (1) and (2), which means that the emergency patient is operated immediately if the emergency OR is empty. Otherwise, the patient has to wait until the emergency OR or one of the elective ORs becomes available.

*Corresponding author
Email address: j.t.vanessen@utwente.nl (J.T. van Essen)

Wullink et al. [6] have shown that for a hospital with 12 ORs the second option is the better one in terms of waiting time, staff overtime, and OR utilization. The waiting time can even be further reduced when the BIMs are spread as evenly as possible over the day. This can be achieved by sequencing the surgeries in their assigned OR, such that the maximum interval between two consecutive BIMs is minimized. We refer to this problem as the BIM problem, which is visualized in Figure 1.

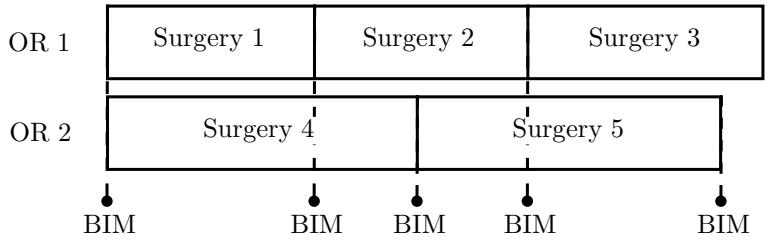

Figure 1: BIM problem

For the BIM problem, there are two situations. In the first situation, each elective surgery is preassigned to an OR. In the second situation, the assignment of elective surgeries to an OR is part of the problem. In this paper, we consider the first situation. This means that we focus on sequencing the elective surgeries in their preassigned OR. The selection of surgeries for a specific OR can, for example, be done using the methods proposed by Lamiri et al. [7] and Gerchak et al. [8]. They propose methods to determine which elective surgeries should be scheduled such that overtime is minimized when a random capacity is needed for emergency surgeries.

As far as we know, the BIM problem is a new type of scheduling problem which has not yet been discussed in literature. A problem quite similar to the BIM problem is the problem Scheduling with Safety Distances (SSD) introduced by Spieksma et al. [9]. The objective of this problem is to schedule the jobs such that the distance between two completion times is larger than a given value d. Therefore, this objective differs from our objective because we aim to minimize the maximum distance between two consecutive completion times.

In Section 2, we give a formal definition of the BIM problem and prove that the BIM problem is strongly NP-hard. We discuss several exact and heuristic solution methods for the BIM problem in Section 3. Computational results for these methods are given in Section 4.

In practice, the initial schedule is disrupted by emergency surgeries and the stochastic durations of elective surgeries. As a result, the completion times of the elective surgeries, and therefore, the BIMs change, leading also to a change of the maximum distance between two BIMs. To investigate the robustness of the initial schedules, we conduct and report a simulation study in Section 5. Section 6 presents conclusions and gives recommendations for further research.

2. Problem Description

In Section 2.1, we formally introduce the BIM problem and in Section 2.2, we prove that the problem is strongly NP-hard.

2.1. Problem Formulation

The BIM problem focuses on sequencing elective surgeries which are assigned to specific ORs. The set of these elective surgeries is given by set $I = \{1, \ldots, M\}$ and each surgery $i \in I$ has an expected duration P_i. We assume that all surgeries have to be scheduled successively with no breaks in between. The set of ORs is given by $J = \{1, \ldots, N\}$ and the set of surgeries that are assigned to OR $j \in J$ is given by $I_j \subset I$. The sets I_j form a partition of set I, and therefore $\sum_j M_j = M$, where M_j denotes the number of surgeries in I_j. Let O_i denote the OR to which surgery $i \in I$ is assigned.

The start time of OR $j \in J$ is denoted by S_j and E_j denotes the moment that all surgeries in OR $j \in J$ are expected to be completed, i.e., $E_j = S_j + \sum_{i \in I_j} P_i$. The interval for which all operating rooms are occupied is called the 'occupied interval'. The start time of the occupied interval is $S = \max_j S_j$ and the end time is given by $E = \min_j E_j$.

We define all moments that can be used to start an emergency surgery as a BIM. These moments include the start and end time of the occupied interval, as well as all completion times of surgeries within the occupied interval. To determine the length of the intervals between two BIMs, the completion times of the surgeries must be ordered in non-decreasing order. We define the interval between two subsequent BIMs as a break-in-interval (BII). Figure 2 visualizes the occupied interval, the BIMs and BIIs for a simplified situation involving two ORs.

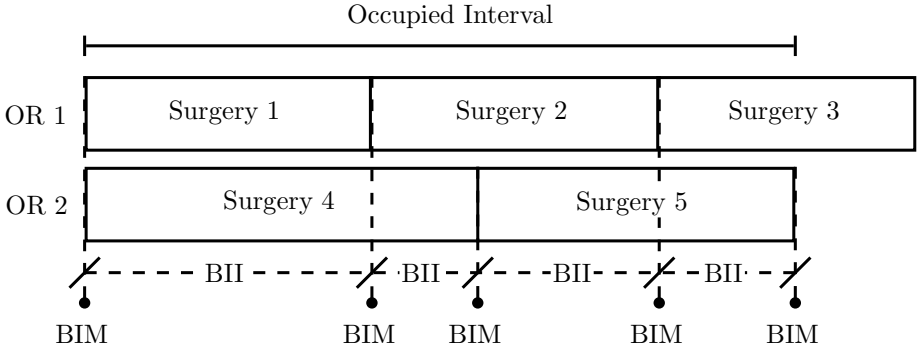

Figure 2: BIM, BII and Occupied Interval

The timing of the BIMs, and thus the length of the BIIs, depends on the sequence of the elective surgeries in each OR. In practice, BIMs appear to be distributed unevenly over the occupied interval, which results in lengthy BIIs that increase the expected waiting time for emergency surgery and therefore, risk medical complications or even mortality. In this paper, we aim to minimize the expected waiting time for emergency surgery by sequencing surgeries in such a way that the BIMs are spread as evenly as possible over the day. We chose to measure 'evenly' by minimizing the maximum BII per day, i.e., min max BII.

By considering the ORs as machines and the surgeries as jobs, we see similarities between the BIM problem and machine scheduling problems. However, our objective of spreading the completion times as evenly as possible has, as far as we know, not previously been treated in the literature. Nevertheless, this objective may occur in several other fields in which walk-in customers have to be served as well as elective procedures.

Another application of the BIM problem in the healthcare sector is to spread the workload on the holding and recovery department. This is the department where patients are prepared before surgery and recover after surgery. When the end times, and therefore also the start times, of surgeries are spread as evenly as possible over the day, the arrivals in and departures from the holding and recovery department are also spread over the day and this improves the work balance for these departments.

2.2. Problem Complexity

Theorem. *The BIM problem is strongly NP-hard for two or more operating rooms.*

Proof. We prove the theorem by reducing 3-partition to the BIM problem. The 3-partition problem can be formulated as follows. Given positive integers $a_1, \ldots, a_{3t}$, and b with $\sum_{j=1}^{3t} a_j = tb$, do there exist t pairwise disjoint subsets $S_i \subset \{1, \ldots, 3t\}$ such that $\sum_{j \in S_i} a_j = b$ for $i = 1, \ldots, t$? The 3-partition problem is proven to be strongly NP-hard (see Garey and Johnson [10]).

The reduction is based on the following transformation. Consider 2 ORs and $5t - 1$ surgeries with the following processing times:

$$\begin{array}{lll} P_i = 4b & \text{for } 1 \leq i \leq t-1, & i \in I_1 \\ P_i = a_{i-t+1} & \text{for } t \leq i \leq 4t-1, & i \in I_1 \\ P_i = 3b & \text{for } 4t \leq i \leq 4t+1, & i \in I_2 \\ P_i = 5b & \text{for } 4t+2 \leq i \leq 5t-1, & i \in I_2. \end{array}$$

First, we notice that the lower bound on our objective min max BII is given by $2b$, since the surgeries of length $4b$ can only be split up by at most one BIM; see Figure 3. Based on this, we prove that the BIM problem has a solution with max BII $= 2b$ if and only if there exists a solution to the 3-partition problem.

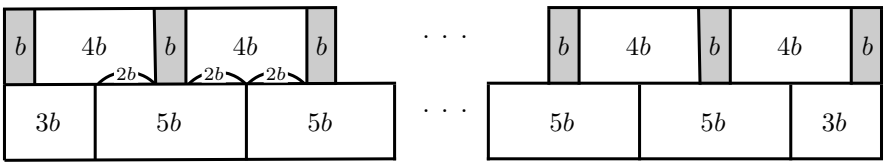

Figure 3: Reduction of 3-partition to the BIM problem

If the 3-partition problem has a solution, we can create an optimal schedule for the BIM problem with $\max \mathrm{BII} = 2b$ as shown in Figure 3. The t blocks of length b correspond to the t subsets for which $\sum_{j \in S_i} a_j = b$.

Furthermore, we show that this is the only way in which a solution with $\max \mathrm{BII} = 2b$ can be created.

- There must be a BIM exactly in the middle of each surgery of length $4b$, since otherwise $\max \mathrm{BII}$ is larger than $2b$. Therefore, a surgery of length $5b$ must start or end in the middle of a surgery of length $4b$. This means that there is a gap of length b between two surgeries each of length $4b$.
- Note that the surgeries of length $3b$ can only be placed at the beginning and end of OR 2 to achieve a solution with $\max \mathrm{BII} = 2b$. This results in a gap of length b at the beginning and end of OR 1.
- Combining the first two points results in a set of t gaps of length b, in which t sets of surgeries with $\sum_{j \in S_i} a_j = b$ have to be scheduled in order to obtain a solution to the BIM problem.

□

3. Solution Methods

In this section, we discuss several solution methods for the BIM problem. In Section 3.1 we give an Integer Linear Program (ILP), which can be used to solve small instances and to benchmark heuristic solution methods. In Section 3.2, we discuss some constructive heuristics, which generate single solutions. Improvement heuristics, which iteratively improve an initial solution, are discussed in Section 3.3. In Section 3.4, some Shifting Bottlenecks Heuristics (SBH) are discussed.

3.1. Exact Solution Method

To solve the BIM problem to optimality, we may model it as an Integer Linear Program (ILP). Based on this model, small instances can be solved using existing ILP solvers. To define proper decision variables, note that a solution to the BIM problem consists of a sequence of surgeries for each OR. These sequences are called 'local' sequences. However, in order to determine the BIIs and thereby the maximum BII, we need to determine the sequence of the BIMs, i.e., the completion times C_i of all surgeries $i \in I$. We call this sequence the 'global sequence'. Figure 4 shows an example of local and global sequences is shown.

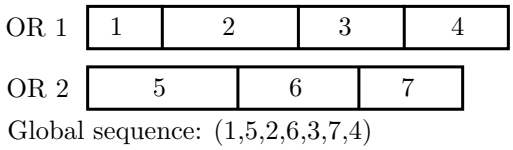

Figure 4: Local sequences and global sequence

The global sequence can be defined by binary variables Y_{ik}, which are one if and only if surgery $i \in I$ is scheduled before surgery $k \in I$ in the global sequence. To ensure that either surgery $i \in I$ is scheduled before surgery $k \in I$, i.e., $Y_{ik} = 1$, or surgery $k \in I$ is scheduled before surgery $i \in I$, i.e., $Y_{ki} = 1$, the following constraint is added:

$$Y_{ik} + Y_{ki} = 1. \tag{1}$$

Based on the values Y_{ik}, the position of surgery $i \in I$ in the global sequence is given by:

$$Z_i = \sum_k Y_{ki}. \tag{2}$$

To ensure that the chosen Y_{ik} values are consistent, we have to ensure that the Z_i's must form a strict totally ordered set. If this is not the case, the Y_{ik} values may contain cycles. To model this, we must have that $Z_i < Z_k$ if $Y_{ik} = 1$, which is ensured by the constraint:

$$Z_i \leq Z_k - 1 + \tilde{M} Y_{ki}, \tag{3}$$

where $\tilde{M}$ is a sufficient large number. Note that this constraint only becomes active when $Y_{ki} = 0$, because when $Y_{ki} = 1$, it does not give any restriction.

A solution to the BIM problem is only feasible, when the completion times resulting from the local sequences are consistent with the global sequence. This means that when surgery $i \in I$ is scheduled before surgery $k \in I$ in the global sequence ($Y_{ik} = 1$), this has to imply that the completion time of surgery $i \in I$, given by C_i, is smaller than or equal to the completion time of surgery $k \in I$, given by C_k. This is ensured by the following constraint:

$$C_i - \tilde{M}(1 - Y_{ik}) \leq C_k, \tag{4}$$

where C_i is given by the start time of OR O_i plus the duration of the surgeries scheduled before surgery $i \in I$ in this OR, including the duration of surgery $i \in I$, i.e.:

$$C_i = S_{O_i} + \sum_{k \in I_{O_i}} P_k Y_{ki}. \tag{5}$$

Note that for two surgeries $i, k \in I$ scheduled in the same OR, constraint (5) already ensures that the values of C_i, C_k and Y_{ik} are compatible.

The BIIs are now given by the difference of the completion times of two consecutive surgeries $i \in I$ and $k \in I$ in the global sequence, i.e., by $C_k - C_i$. Surgeries $i \in I$ and $k \in I$ are scheduled consecutively when surgery $i \in I$ is scheduled immediately before surgery $k \in I$ in the global sequence, i.e., when $Z_k - Z_i = 1$. To model this, we introduce binary variables D_{ik}, which take value one if $Z_k - Z_i \leq 1$. This is ensured by constraint (6). The variables D_{ik} are used to only include the positive BIIs of consecutive BIMs when determining the maximum BII (see constraint (8)):

$$Z_k - Z_i \geq 2 - \tilde{M} D_{ik}. \tag{6}$$

Note that $D_{ik} = 1$ if either surgery $k \in I$ is scheduled directly after or somewhere before surgery $i \in I$ in the global sequence. In the latter case, $C_k - C_i$ is negative. However, this will not cause any problems, because the maximum BII is at least zero.

A BIM should not be taken into account when it lies outside the occupied interval. To model this, we introduce binary variables $\tilde{S}_i$, which have to take value one if the completion time of surgery $i \in I$ is larger than or equal to the start time of the occupied interval, i.e., when $C_i \geq S$. Binary variables $\tilde{E}_i$ have to take value one if the completion time of surgery $i \in I$ is smaller than or equal to the end time of the occupied interval, i.e., when $C_i \leq E$. This is ensured by the following two constraints, where ϵ is a sufficiently small number:

$$\begin{aligned} C_i &\geq E + \epsilon - \tilde{M}\tilde{E}_i, \\ C_i - P_i &\leq S - \epsilon + \tilde{M}\tilde{S}_i. \end{aligned} \tag{7}$$

Using the variables D_{ik}, $\tilde{E}_i$, and $\tilde{S}_i$, we now can model the objective to minimize the maximum BII by introducing a variable H, which denotes the maximum BII, and for which the following constraint must hold:

$$H \geq C_i - C_k - \tilde{M}(1 - D_{ik}) - \tilde{M}(1 - \tilde{E}_i) - \tilde{M}(1 - \tilde{S}_k) \quad \forall i, k \in I. \tag{8}$$

This constraint states that H should be greater than or equal to each BII of two consecutive surgeries in the global sequence, when their completion times lie inside the occupied interval. The variable D_{ik} guarantees that the only positive BIIs considered for H are those between two consecutive BIMs (note that when $Z_i - Z_k > 1$, the variable D_{ik} is equal to 0). The variables $\tilde{S}_k$ and $\tilde{E}_i$ guarantee that only BIMs within the occupied interval are considered, because when $C_i > E$ or $C_k < S$, the variable $\tilde{E}_i$ or $\tilde{S}_k$ is equal to 0. The objective function is now given by $\min H$. Summarizing, the resulting ILP for the BIM problem is:

$$\begin{aligned} &\min H \\ &\text{s.t. (2)-(8)} \\ &Y_{ik}, D_{ik}, \tilde{S}_i, \tilde{E}_i \in \{0,1\} \quad \forall i,k \in I. \end{aligned} \tag{9}$$

This ILP consists of $2M^2 + 4M + 1$ variables and $4M^2 + 4M$ constraints, where M is the number of surgeries. The $2M^2 + 4M + 1$ variables can be divided into $2M^2 + 2M$ binary variables and $2M + 1$ continuous variables.

In practice, we may need to add more constraints to this ILP. Some surgeries may have to be performed early in the day, for example surgeries on children, while other surgeries may have to be performed at the end of the day, for example when extra cleaning is needed after surgery. We also have to take into account the availability of scarce resources, such as microscopes or X-ray machines. In addition, the surgeon may have a preference for the sequence of surgeries. Many of these constraints can be incorporated in the ILP model in a straightforward way.

3.2. Constructive Heuristics

As the ILP model of the previous subsection can only be used to solve small instances, in this subsection, we discuss four constructive heuristics which can be used to generate a single solution. If there are ORs that start before the occupied interval (i.e., if not all ORs start at the same time), we perform a preprocessing step in which some surgeries are removed. These are the surgeries that are scheduled outside the occupied interval in the preprocessing step, which means that both the start and completion time of the surgery lie before the start time or after the end time of the occupied interval. We give preference to remove lengthy surgeries, because short surgeries may help more to decrease the maximum BII.

Step 1. Schedule the surgeries of each OR by Longest Processing Time first (LPT). Let $I^1 \subset I$ be the set of surgeries for which the completion time C_i in this LPT schedule is smaller than or equal to the start time of the occupied interval, i.e., for which $C_i \leq S$. Delete these surgeries from set I and I_j, and update the start time of the OR, i.e. $I \setminus I^1$, $I_j \setminus I_j^1$ and $S_j = \max_{i \in I_j^1} C_i$.

Step 2. Schedule the remaining surgeries of each OR by SPT. Let $I^2 \subset I$ be the set of surgeries for which the start time $C_i - P_i$ in this SPT schedule is greater than or equal to the end time of the occupied interval, i.e., for which $C_i - P_i \geq E$. Delete these surgeries from set I and I_j, and update the end time of each OR, i.e. $I \setminus I^2$, $I_j \setminus I_j^2$ and $E_j = \min_{i \in I_j^2} C_i - P_i$.

After this preprocessing step, we have at least one surgery for each OR that can be scheduled first or last such that it partly lies in the occupied interval. However, there still may exist surgeries with a short duration that lie completely outside the occupied interval when scheduled first or last in their OR.

The first heuristic considered is based on list-scheduling and specific on Shortest Processing Time first (SPT). Furthermore, we propose three other heuristics that aim to sequence the surgeries such that every BII approaches a lower bound λ on the max BII, which is given as follows:

$$\lambda = \frac{E - S}{1 + \sum_{j \in J} (M_j - 1)} = \frac{E - S}{1 + M - N}. \tag{10}$$

This lower bound reflects the interval between two BIMs if all completion times are evenly distributed in the occupied interval O. The length of the occupied interval, given by $E - S$, is divided by the number of expected

BIIs. For each OR the number of expected BIIs equals the number of surgeries assigned to this OR given by M_j. However, in this way, we count the end of the occupied interval N times while we only have to count it once. Therefore, the number of expected BIIs is given by $1 + M - N$. Note that this bound is one of the reasons to do the preprocessing step, because else $1 + M - N$ surely would be an overestimate of the expected number of BIMs in the occupied interval.

3.2.1. List Scheduling

The list scheduling algorithm sorts the surgeries in some order and then, iteratively, adds surgeries to the schedule in this order. We add a quality constraint, which ensures that the first entry on the list is only added if it does not violate this quality constraint. Otherwise, the next surgery on the list that does not violate the quality constraint is scheduled. If there is no such entry, the last surgery on the list is scheduled. The quality constraint is defined as follows. Let λ be defined as in (10) and let τ be the length of the BII that will be created by the considered surgery; see Figure 5. The quality constraint is defined as follows:

$$\tau \geq \lambda(1 - \beta).$$

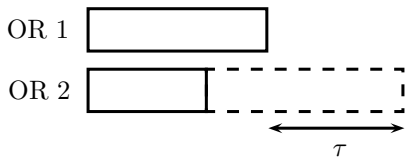

Figure 5: SPT with τ

This means that the created BII should not be smaller than λ minus a factor β of λ with $0 < \beta < 1$, i.e., we only accept a new BII if it is relatively close to its goal value. Next to the quality constraint, we also guarantee that the BIMs occur in non-decreasing order. This means that scheduling a surgery $i \in I$ in OR $O_i \in J$ is only allowed if for all other ORs still a surgery exists which leads to a BIM larger than or equal to the completion time of surgery $i \in I$.

Within the list scheduling algorithm, every possible order of surgeries may be used. We have chosen the SPT order, which sorts all surgeries in non-decreasing order of processing times. The resulting algorithm is called the SPT-algorithm.

3.2.2. Fixed Goal Values

The 'Fixed Goal Values' constructive heuristic has some similarities to the list scheduling algorithm, which is described above. It also builds a schedule in which the BIIs try to approximate the goal value λ as good as possible. However, now no pre-specified order of the surgeries is given, but in each step, a surgery is selected for which the new created BII approximates λ the best. We start at the beginning of the day and schedule the surgery for which the completion time approximates $S + \lambda$ the best. Next, we choose from the remaining surgeries the surgery for which its completion time best approximates $S + 2\lambda$. We continue until all surgeries are scheduled. As for list scheduling, we again guarantee that the BIMs occur in non-decreasing order. The specific description of this constructive heuristic is given below. Here, B_j is the new start time of OR j resulting from surgeries scheduled in previous iterations.

Step 1. Determine λ, set $t = 1$, $I' = I$, $I'_j = I_j$ and $B_j = S_j$.

Step 2. Determine for each unscheduled surgery $i \in I'$ the completion time if it is scheduled next, i.e., $C_i = B_{O_i} + P_i$.

Step 3. Let $\bar{I} \subseteq I'$ be the set of unscheduled surgeries $i \in I'$ with $C_i \leq \min_{j \in J} \left(B_j + \max_{k \in I'_j} P_k \right)$. The restriction to the subset $\bar{I}$ is needed to guarantee that the sequence of BIMs is non-decreasing.
Set $k = \arg\min_{i \in \bar{I}} |\lambda t - C_i|$.

Step 4. Determine for OR O_k the new start time, i.e., $B_{O_k} = B_{O_k} + P_k$. Delete surgery $k \in I$ from set I' and I'_{O_k}, i.e., $I' = I' \setminus \{k\}$ and $I'_{O_k} = I'_{O_k} \setminus \{k\}$. Set $t = t + 1$.

Step 5. If $I' = \emptyset$ then stop, else go to Step 2.

The goal value λ is calculated once at the beginning of the algorithm. However, it may happen that the scheduled surgeries always under- or overestimate the goal value λ. Therefore, it may be beneficial to let λ depend on the already scheduled surgeries. The following subsection describes a heuristic that incorporates this suggestion.

3.2.3. Flexible Goal Values

The idea of 'Flexible Goal Values' is similar to the 'Fixed Goal Values' algorithm. However, in this heuristics, the value of λ is updated in each step. After a surgery is scheduled, we determine the new occupied interval which starts at the completion time of the last scheduled surgery. In addition, the number of surgeries to be scheduled, is decreased by one. Let B denote the new start of the occupied interval and K the number of scheduled surgeries, then λ is given by:

$$\frac{E - B}{1 + M - N - K}. \tag{11}$$

The specific description of this constructive heuristic is given by the following steps.

Step 1. Set $I' = I$, $I'_j = I_j$, $B_j = S_j$, and $K = 0$.
Step 2. Set $B = \max_{j \in J} B_j$ and determine λ.
Step 3. Determine for each surgery $i \in I'$ the completion time if it is scheduled next, i.e., $C_i = B_{O_i} + P_i$.
Step 4. Let $\bar{I} \subseteq I'$ be the set of unscheduled surgeries $i \in I'$ with $C_i \leq \min_{j \in J} \left(B_j + \max_{k \in I'_j} P_k \right)$. The restriction to the subset $\bar{I}$ is needed to guarantee that the sequence of BIMs is non-decreasing.
Set $k = \arg\min_{i \in \bar{I}} |B + \lambda - C_i|$.
Step 5. Determine for OR O_k the new start time, i.e., $B_{O_k} = B_{O_k} + P_k$. Delete surgery $k \in I$ from set I' and I'_{O_k}, i.e., $I' = I' \setminus \{k\}$ and $I'_{O_k} = I'_{O_k} \setminus \{k\}$. Set $K = K + 1$.
Step 6. If $I' = \emptyset$ then stop, else go to Step 2.

It may be worthwhile to apply this heuristic not only in a forward manner but also in a backward manner starting at the end time E of the occupied interval. The following subsection describes a heuristic that incorporates this suggestion.

3.2.4. Forward-Backward Scheduling

The 'Forward-Backward' algorithm builds further on the 'Flexible Goal Values' algorithm. However, in this heuristic, we schedule both forwards and backwards. Forward scheduling is the scheduling of surgeries one after another from the start of the day (S_j) towards the end of the day (E_j), while the reverse holds for backward scheduling. Backward scheduling is possible, since the number and the durations of surgeries in an OR are known. For the 'Flexible Goal Values' algorithm, it may happen that the lengthy surgeries are scheduled at the end of the day, which may result in large BIIs. By scheduling forwards and backward, we try to avoid this. As in the 'Flexible Goal Values' algorithm, the value of λ is updated in each step by adjusting the start and end times of the occupied interval. The adjusted start time of the occupied interval is denoted by B, and the adjusted end time of the occupied interval is denoted by F. Then, λ is given by:

$$\frac{F - B}{1 + M - N - K}. \tag{12}$$

The specific description of this constructive heuristic is given as follows.

Step 1. Set $I' = I$, $I'_j = I_j$, $B_j = S_j$, $F_j = E_j$, and $K = 0$.
Step 2. Set $B = \max_{j \in J} B_j$, $F = \min_{j \in J} F_j$, and determine λ.
Step 3. Determine for each surgery $i \in I'$ the completion time if it is scheduled next forward, i.e., $C_i = B_{O_i} + P_i$, and if it is scheduled backward, i.e., $\tilde{C}_i = F_{O_i}$.

Step 4. Let $\bar{I}_f \subseteq I'$ be the set of unscheduled surgeries $i \in I'$ with $C_i \leq \min_{j \in J} \left(B_j + \max_{k \in I'_j} P_k \right)$. The restriction to the subset $\bar{I}_f$ is needed to guarantee that the sequence of BIMs is non-decreasing.
Let $\delta_f = \min_{i \in \bar{I}} |B + \lambda - C_i|$ and set $k_f = \arg\min_{i \in \bar{I}} |B + \lambda - C_i|$.
Let $\bar{I}_b \subseteq I'$ be the set of unscheduled surgeries $i \in I'$ with $\tilde{C}_i - P_i \geq \max_{j \in J} \left(F_j - \max_{k \in I'_j} P_k \right)$. The restriction to the subset $\bar{I}_b$ is needed to guarantee that the sequence of BIMs is non-decreasing.
Let $\delta_b = \min_{i \in \bar{I}} |F - \lambda - (\tilde{C}_i - P_i)|$ and set $k_b = \arg\min_{i \in \bar{I}} |F - \lambda - (\tilde{C}_i - P_i)|$.

Step 5. If $\delta_f < \delta_b$, then $k = k_f$, else $k = k_b$. Determine for OR O_k the new start or end time, i.e., if $\delta_f < \delta_b$, then $B_{O_k} = B_{O_k} + P_k$, else $F_{O_k} = F_{O_k} - P_k$. Delete surgery $k \in I$ from set I' and I'_{O_k}, i.e., $I' = I' \setminus \{k\}$ and $I'_{O_k} = I'_{O_k} \setminus \{k\}$. Set $K = K + 1$.

Step 6. If $I' = \emptyset$ then stop, else go to Step 2.

3.3. Improvement Heuristics

The constructive heuristics presented above all have a local view when scheduling the next surgery and do not consider the 'global' consequences of the decisions. Therefore, the resulting schedule may leave room for improvement. An important class of heuristics to achieve such improvement is that of local search procedures. However, a local search procedure does not guarantee an optimal solution. It attempts to find a better solution in the neighborhood of the current solution and repeats the procedure in an iterative manner. The neighborhood of the current solution is defined as a set of solutions that can be obtained through an allowed modification of the current solution. For the local search procedures described in this section, we use a 2-exchange neighborhood. In this neighborhood, a solution is a neighbor when it can be achieved by exchanging two surgeries in one OR. Note that exchanging surgeries from two different ORs is not allowed. We also experimented with other neighborhood structures [11], but the 2-exchange neighborhood performed the best in preliminary tests. In each iteration of the local search procedure, a solution from the neighborhood is generated and evaluated. The procedure either accepts or rejects this candidate solution as the next solution to move on to, based on a given acceptance-rejection criterion. In the following two subsections, we discuss two types of local search procedures, namely Simulated Annealing (SA) and Tabu Search (TS).

Before we go into detail, we first introduce a change in the way the objective function is defined. If we only minimize the maximum BII, two solutions are considered to be equally good if the length of the maximum BIIs are the same. However, the other BIIs also influence the quality of a solution, because we aim to spread the BIMs as evenly as possible. This means that when the maximum BIIs of two solutions are of equal length, and the second largest BII of one solution is smaller than the second largest BII of another solution, the first solution is better. Preliminary tests have shown that it is sufficient to consider only the three largest BIIs. Therefore, we introduce a new objective function which is a lexicographic vector of the three largest BIIs. Furthermore, note that the constructive heuristics, due to their description, already focus on all BIIs.

3.3.1. Simulated Annealing

The name of and inspiration for Simulated Annealing (SA) [12] originate from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce defects. The heat causes the atoms to move from their initial positions to other random states of higher energy. The slow cooling gives them more chance of finding configurations with lower internal energy than the initial one.

By analogy with this physical process, each step of SA possibly moves from the current solution to a randomly selected neighbor solution. If the neighbor solution has a lower objective function value than the current solution, the neighbor solution is accepted. Otherwise, the neighbor solution is accepted with probability $e^{\frac{-\Delta}{T}}$, where Δ is the difference between the objective value of the current and the neighbor solution, and T is the current temperature. If the neighbor solution is rejected, SA stays at the current solution. Note that the probability of accepting a worse solution is proportional to the difference in objective values since $e^{\frac{-\Delta}{T}} \sim 1 - \frac{\Delta}{T}$. Slightly worse solutions have a reasonably high probability of being accepted, while much worse solutions are only accepted infrequently. The temperature T gradually decreases during the search process and therefore also the acceptance probability of a worse solution decreases. The allowance of moving to worse solutions makes it possible to escape from a (poor) local minimum.

To apply SA, the stopping criterion and cooling scheme must be specified. The cooling scheme needs to specify the initial value of T and how this temperature is updated in each step of the procedure. The initial temperature is normally chosen such that in the first iterations most of the neighbor solutions have a relatively high probability of being accepted. The updating of the temperature T is often done by decreasing it in every iteration by a factor α with $0 < \alpha < 1$. For each temperature level, we perform several iterations which form a Markov chain, because the next state only depends on the current state. The length of the Markov chain, indicated by L, should be chosen such that it equals the size of the neighborhood [13]. As stopping criterion, we set a threshold for the temperature. This threshold T_f is chosen such that worse solutions have a low probability of being accepted. SA can be summarized by the following algorithm, where $\bar{S}$ denotes the current best solution.

Step 1. Generate an initial solution S using some heuristic and determine its objective function vector $\mathbf{f}(S)$. Set $\bar{S} = S$. Set an initial temperature T and a reduction factor α.
Step 2. Repeat L times:
  Step a) Select a neighbor solution S' of solution S at random and determine $\mathbf{f}(S')$.
  Step b) If $\mathbf{f}(S') \leq \mathbf{f}(S)$, set $S = S'$, and if $\mathbf{f}(S') \leq \mathbf{f}(\bar{S})$, set $\bar{S} = S'$.
    If $\mathbf{f}(S') > \mathbf{f}(S)$, set $S = S'$ with probability $e^{\frac{-\Delta}{T}}$.
Step 3. Set $T = \alpha T$. If $T < T_f$, then stop; else, go to Step 2.

In Section 4, we give a description of the stopping criterion and cooling scheme used for our computational results.

3.3.2. Tabu Search

Like SA, Tabu Search (TS) [12] moves from one solution to another with the new solution being possibly worse than the one before. The main difference between TS and SA is the mechanism used for accepting a candidate solution. In TS, the mechanism is not probabilistic but deterministic, because it systematically searches the neighborhood and selects the best solution found, even if this solution is worse than the current solution. During the process, a tabu list is kept which contains solutions, or properties of the solutions, the heuristic is not allowed to accept. In the most simple case, the tabu list has a fixed number of entries. Every time a new solution is accepted, the current solution, or some property of the current solution, enters the tabu list and the oldest entry is deleted. The tabu list aims to avoid returning to a local minimum that has been visited previously. TS can be summarized by the following algorithm, in which $\bar{S}$ is the current best solution.

Step 1. Generate an initial solution S using some heuristic and determine its objective function value, $\mathbf{f}(S)$. Set $\bar{S} = S$.
Step 2. Select the neighbor solution S' of solution S that has the lowest objective function value, and that is not tabu. Solution S', or a property of solution S', enters the tabu list and the oldest entry is deleted. Set $S = S'$.
Step 3. If $\mathbf{f}(S') \leq \mathbf{f}(\bar{S})$, then set $\bar{S} = S'$.
Step 4. If the stopping criterion is met, then stop. Else, go to Step 2.

In Section 4, we define the length of the tabu list and give a description of the used solution property.

3.4. Shifting Bottleneck Heuristics

Besides pure constructive or improvement heuristics, also some combined approaches exist. The Shifting Bottleneck Heuristic (SBH) [14] is such a combination. In each step of the procedure, the OR that is considered to be the bottleneck is selected. For this OR, a sequence of the surgeries is determined. After this, (a subset of) all already scheduled ORs are rescheduled. This can be seen as an improvement step. In the following paragraphs, the three steps of SBH are further explored.

The first step in our SBH is the selection of the bottleneck OR. This is the OR that may give the most problems in minimizing the maximum BII. We consider three options, namely (1) the OR with the smallest number of surgeries, (2) the OR to which the longest surgery is assigned, and (3) the OR that has the longest average surgery duration. The first option selects the OR that creates the smallest number of BIMs, and therefore, contributes the least to creating small BIIs. An advantage of this option is that an OR with a small number of surgeries can be

rescheduled faster than an OR with a large number of surgeries. This can reduce the overall run time, because ORs that are added early in the process are rescheduled more often. The second option is considered because long surgeries can result in large BIIs. However, following the same reasoning we can choose the OR with the longest average surgery duration, the third option. Preliminary results have shown that, in most cases, the third option results in a shorter runtime and smaller BIIs.

The next step is to determine a schedule for the bottleneck OR. As in practical instances there are only a few surgeries scheduled in each OR, we have chosen to enumerate all possible sequences of the surgeries. For each of these sequences, we determine the three largest BIIs with respect to the BIMs of already scheduled ORs. We choose the sequence that creates the lexicographic smallest three largest BIIs.

The final step is to reschedule some already scheduled ORs to reduce the three largest BIIs. Note that the maximum BII can not only be reduced by resequencing the surgeries of the OR which contains this maximum BII, but also by resequencing the surgeries of other ORs. This resequencing can split up the maximum BII by scheduling a BIM during this BII. We choose to reschedule the OR for which the completion time of one of the preassigned surgeries creates the start or end time of the smallest BII. The reasoning for this choice is that we hope to reduce the maximum BII by increasing the smallest BII. The selected OR is rescheduled by using the same technique as in the previous step. Note that this rescheduling method can also be used as an improvement heuristic for the constructive heuristics.

4. Computational Results

We have tested the heuristics described in Section 3 on several instances. These instances are created based on information derived from 10 years of empirical data of the OR department for inpatients in the Erasmus Medical Center. The number of ORs varies from 2 to 16, and all ORs are assumed to be generic as opposed to be dedicated to a specialty, i.e., each emergency surgery can be scheduled in all ORs. Each instance consists of data for 100 periods of 5 days and all ORs start at the same time. The solution methods are implemented in AIMMS 3.10 and run on an Intel Core2 Duo CPU P8600 2.40 GHz with 3.45 GB RAM. The parameter settings for SA and TS are given in Subsection 4.1. In Subsection 4.2, we discuss the obtained results.

4.1. Parameter Settings

To use the developed heuristics in practice, the runtime should be low. Therefore, we limit the number of iterations for both SA and TS. The parameter settings of SA and TS, which implicitly define the number of iterations, are given in the following subsections.

4.1.1. Simulated Annealing

For SA, we choose as initial solution the solution given by the 'Flexible Goal Values' (FGV) algorithm because FGV is the best constructive heuristic (see Subsection 4.2). The initial temperature is chosen such that a worse solution compared to FGV is accepted with at least probability 0.5 (see Busetti [15]). We establish this by basing the initial temperature on the upperbound Λ of the objective function value given by:

$$\Lambda = \min_{j \in J} \max_{i \in I_j} P_i. \tag{13}$$

This upperbound occurs when the largest surgeries of each OR are scheduled simultaneously, see Figure 6.
This leads to the initial temperature which is given by:

$$T_0 = \frac{f(\text{FGV}) - \Lambda}{\ln 0.5}.$$

We set the length of the Markov chain equal to half the size of the neighborhood structure. On average we have 5 surgeries in each OR and therefore, we set the length of the Markov chain to $\frac{1}{2}N\binom{5}{2}$ which represents half the number of possible 2-exchanges. We set the reduction factor α to 0.85.

As stopping criterion, we use a threshold for the temperature. We define this threshold such that the probability of accepting a neighbor solution S' with $\mathbf{f}(S') \geq \mathbf{f}(S) + 20$, which is a regular occurring change in the objective value, is less than 0.01. Therefore, the final temperature is given by:

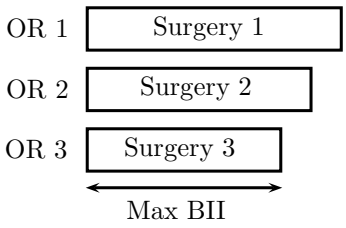

Figure 6: Upperbound

$$T_f = \frac{-20}{\ln 0.01}.$$

This cooling scheme leads to approximately 600, 1000, and 1800 iterations for 6, 10 and 16 ORs respectively.

4.1.2. Tabu Search

As initial solution for TS, we also choose the solution given by FGV and we set the length of the tabu list to 12. We also have to determine what property of the solution is included in the tabu list. A feasible solution is uniquely defined by the start times of each surgery. However, if we would choose this option it takes a lot of time to determine whether a solution is tabu or not. Therefore, the tabu list should not contain the solution, but some property of the solution. One candidate property is the objective function value of the solution. However, there exists many different solutions with the same objective value, and therefore, no valuable information is obtained by using this property. Another candidate property is the place of the two exchanged surgeries in the local sequence. However, the start times of the two exchanged surgeries provide much more information because it implicitly provides some information about the sequence of the other surgeries in the considered OR. A drawback is that no information is available about the sequence in the other ORs, but providing this information will require much more memory. By using the start time of the exchanged surgeries after they are exchanged, it may happen in theory that TS returns to an already visited solution, however, this is not likely to happen in practice.

When a new best solution is found, we reset the tabu list. This helps TS to fully search a new neighborhood. The process stops when after 12 iterations no new best solution is found. This stopping criterion and the length of the tabu list lead to approximately the same number of iterations for SA and TS.

4.2. Results for the Instances

The schedules created by our solution methods are based on the expected duration of the elective surgeries. For each instance and solution method, we determine the average runtime and the average maximum BII over the 500 days. We also display the standard deviation of the maximum BII. We evaluate the quality of the heuristic solution methods by the lowerbound for the maximum BII, which is given by $\max(\lambda, \min_{i \in I} P_i)$. In addition, we display the average maximum BII and standard deviation of the original schedules that were used in the Erasmus Medical Centre to see if the proposed solution methods find a smaller maximum BII. To determine the effect of SA and TS, we also randomly generated approximately the same number of solutions as generated by SA and TS. From these solutions, we choose the solution which minimizes the maximum BII. Tables 1-3 show the results for 6, 10, and 16 ORs.

The results show that all heuristic solution methods, except SPT, provide a better solution than the original schedule used at the Erasmus Medical Centre. The average length of the maximum BII decreased with more than 20%. The best heuristic method is TS for which the maximum BII decreased with more than 40%. Although its average runtime is longer than the runtime of the constructive heuristics, it is still very fast (less than one second). The gap between the lowerbound and the objective value of TS decreases when the number of ORs increases. This is expected because it is easier to generate BIIs close to the lowerbound λ, when the number of surgeries increases.

By using the ILP formulation, we can only solve the instance with 2 ORs to optimality within an acceptable amount of time. The average maximum BII equals 84.9 minutes, which is less than 4% lower than the average maximum BII of TS. Therefore, it seems that TS performs relatively well.

	Avg. Runtime (s)	Avg. Max BII (min)	Std Dev (min)
Original Schedule	—	66.2	19.5
Lowerbound	—	29.4	9.8
SPT	0.0	58.2	13.7
Fixed Goal Values	0.0	46.8	10.9
Flexible Goal Values	0.0	45.2	10.8
Forward-Backward	0.0	48.3	12.9
Random	—	38.9	5.4
SA	0.2	38.7	6.1
TS	0.2	36.6	7.6
SBH	0.3	38.7	7.3

Table 1: Instance with 6 ORs

	Avg. Runtime (s)	Avg. Max BII (min)	Std Dev (min)
Original Schedule	—	48.1	10.7
Lowerbound	—	25.6	8.0
SPT	0.0	46.8	8.7
Fixed Goal Values	0.0	35.0	8.3
Flexible Goal Values	0.0	33.2	7.6
Forward-Backward	0.0	37.5	9.3
Random	—	29.0	6.6
SA	0.3	28.9	6.9
TS	0.3	26.8	7.7
SBH	0.8	28.5	7.6

Table 2: Instance with 10 ORs

	Avg. Runtime (s)	Avg. Max BII (min)	Std Dev (min)
Original Schedule	—	39.1	7.8
Lowerbound	—	22.7	4.9
SPT	0.1	41.6	7.1
Fixed Goal Values	0.0	26.3	6.2
Flexible Goal Values	0.0	25.4	5.8
Forward-Backward	0.1	31.5	9.5
Random	—	22.8	4.9
SA	0.7	22.8	4.9
TS	0.6	22.7	4.9
SBH	1.0	22.9	5.0

Table 3: Instance with 16 ORs

When we compare TS to the random solution, we note that there is only a small difference between the average maximum BIIs. This difference is even smaller for SA. Therefore, we conclude that SA does not have a large effect when we want to minimize the maximum BII.

The 'Flexible Goal Values' algorithm is the best constructive heuristic. We might expect that the 'Forward-Backward' algorithm would perform better, but by scheduling forwards and backwards, the algorithm leaves a gap in the middle which results in a large maximum BII. SPT is the worst constructive heuristic. This is what we expect, because SPT does not explicitly consider the objective of the BIM problem.

5. Simulation Results

In practice, emergency patients arriving throughout the day and a change in the duration of elective surgeries may disrupt the initial schedule. With the use of simulation, we want to quantify the change in the objective function value of the created schedules due to these arrivals and the surgery duration variability. We model the arrival of emergency surgeries as a Poisson process, i.e., the interarrival times between emergency surgeries are independent and exponentially distributed. Furthermore, the duration of the emergency surgeries are chosen from a lognormal distribution. The parameters of the distributions are calculated based on the data of the Erasmus Medical Centre.

The emergency surgeries are scheduled first-come-first serve at the first available BIM after their arrival. All subsequent elective surgeries are postponed, which may result in overtime. Elective surgeries cannot start earlier than their scheduled time, because patients may not be ready for surgery. The durations of the elective surgeries used in the simulation study are generated from a lognormal distribution based on the data of the Erasmus Medical Centre.

	Expected Avg. Max BII (minutes)	Simulated Avg. Max BII (minutes)	Simulated Std Dev BII (minutes)	Average Wait Time (minutes)	% > 60 min	% > 30 min
Original Schedule	66.2	67.9	19.1	24.4	2.7	13.7
SPT	58.2	63.5	17.3	23.2	2.3	12.2
Fixed Goal Values	46.8	61.3	16.5	21.1	1.4	10.5
Flexible Goal Values	45.2	61.3	16.3	21.1	1.5	10.7
Forward-Backward	48.3	61.7	17.0	21.6	1.6	10.9
Random	38.9	62.6	16.8	21.7	1.6	11.1
SA	38.7	62.2	16.6	21.5	1.5	10.8
TS	36.6	61.8	16.5	21.1	1.4	10.6
SBH	38.7	62.2	16.6	21.5	1.5	10.8

Table 4: Instance with 6 ORs

For each evaluated schedule, the number of simulation runs is set to 200. With this value, we obtain a minimal confidence interval of 95%. We do not only determine the new average maximum BII and the standard deviation denoted by 'Simulated Avg. Max BII' and 'Simulated Std Dev BII', but also the average waiting time of the emergency surgeries. In addition, we measure the percentage of emergency surgeries for which the waiting time is longer than 60 minutes and 30 minutes. The objective function values of the solution methods discussed in the previous section are shown in column 'Expected Avg. Max BII'. Tables 4-6 show the results for 6, 10, and 16 ORs.

The results show that after simulation, all heuristic methods, except SPT, provide a better solution than the original schedule in terms of maximum BII and waiting time. Therefore, it is beneficial to consider the maximum BII criteria when creating a schedule. However, the simulated average maximum BII is much larger than the expected average maximum BII as a result of the disturbances. Therefore, replanning may be needed if the disturbances of the initial schedule get too large. The simulated average maximum BII and waiting time are similar for all heuristics solution methods, except for SPT. This also holds for the percentage of emergency patients who have to wait more than 60 or 30 minutes.

The simulation study shows that the 'Fixed Goal Values' algorithm performs the best, because both the average waiting time and average maximum BII are the lowest. The 'Fixed Goal Values' algorithm not only focuses on minimizing the three largest BIIs, but also on creating BIIs with approximately equal lengths, which means that the BIMs are spread more evenly over the day. Because of this, a more stable schedule is generated which better adjusts to changes. TS, for example, only focuses on minimizing the three largest BIIs and this can result in a relatively high variance of the length of the other BIIs. The second best heuristic is the 'Flexible Goal Values' algorithm, followed by TS.

The simulation results also show that the average maximum BII and average waiting time decrease when the number of ORs increases. As stated in the previous section, this can be explained by a decrease in the lowerbound

	Expected Avg. Max BII (minutes)	Simulated Avg. Max BII (minutes)	Simulated Std Dev BII (minutes)	Average Wait Time (minutes)	% > 60 min	% > 30 min
Original Schedule	48.1	48.6	12.4	11.8	0.2	3.5
SPT	46.8	48.7	12.8	12.9	0.5	4.1
Fixed Goal Values	35.0	43.5	10.9	9.8	0.1	1.9
Flexible Goal Values	33.2	43.7	10.5	10.0	0.1	2.0
Forward-Backward	37.5	43.6	10.9	10.4	0.1	2.3
Random	29.0	44.7	11.1	10.3	0.1	2.2
SA	28.9	44.4	11.0	10.1	0.1	2.1
TS	26.8	43.9	10.6	10.0	0.1	2.0
SBH	28.5	44.5	11.1	10.2	0.1	2.2

Table 5: Instance with 10 ORs

	Expected Avg. Max BII (minutes)	Simulated Avg. Max BII (minutes)	Simulated Std Dev BII (minutes)	Average Wait Time (minutes)	% > 60 min	% > 30 min
Original Schedule	39.1	36.0	8.5	5.5	0.0	0.7
SPT	41.6	39.5	10.6	7.6	0.1	1.6
Fixed Goal Values	26.3	31.4	7.5	4.1	0.0	0.2
Flexible Goal Values	25.4	31.7	7.3	4.2	0.0	0.2
Forward-Backward	31.5	31.9	7.2	4.8	0.0	0.3
Random	22.8	32.5	7.6	4.5	0.0	0.3
SA	22.8	32.4	7.6	4.4	0.0	0.3
TS	22.7	31.9	7.3	4.2	0.0	0.2
SBH	22.9	32.5	7.5	4.5	0.0	0.3

Table 6: Instance with 16 ORs

λ. The difference between the expected and simulated average maximum BII also decreases with the number of ORs.

6. Conclusions and Recommendations

We have introduced the BIM problem that deals with sequencing elective surgeries, which are preassigned to an OR, to reduce emergency surgery waiting time. As far as we know, this is a new type of scheduling problem which has not yet been discussed in literature. We developed and tested several heuristic solutions methods to solve this new BIM problem.

The results of the computational study show that the expected maximum BII can be reduced by more than 20%. The simulation results show that it is beneficial to take the maximum BII into account when creating a schedule, because our heuristics outperform the original schedule. The best expected solution is provided by TS. However, the 'Fixed Goal Values' algorithm provides the best simulated solution. Therefore, we advice hospitals to use the 'Fixed Goal Values' algorithm because it can easily be implemented and provides good solutions. The reason that the 'Fixed Goal Values' algorithm provides a better solution than TS is that TS only focuses on minimizing the three largest BIIs.

The 'Fixed Goal Values' algorithm can also be used to spread the workload of the holding and recovery department. This is the department in which patients are prepared before surgery and recover after surgery. When the completion times - and therefore also the start times - of surgeries are spread as evenly as possible over the day, the arrivals in and departures from the holding and recovery department are also spread over the day.

Although it may improve the results a lot, we do not consider on-line rescheduling, because in practice, it is often, for example, not possible to exchange two elective surgeries during the day. Even scheduling an elective surgery earlier in the day may not be allowed, because patients may not have arrived at the hospital yet. This means that elective surgeries can only be postponed or cancelled. Another option to reduce this negative effect of disturbances is to insert breaks into the schedule, see Hartholt [16].

Further research should focus on implementing practical constraints. So far, we have not considered the availability of scarce resources such as microscopes or X-ray machines. In addition, the sequence of surgeries might be restricted because, for example, children should be scheduled early in the day. We also assumed that emergency surgeries can be performed in any of the available ORs, and should always be performed as soon as possible. However, in practice, some of the emergency surgeries can only be performed in a subset of the ORs which have specialised resources, and some surgeries are more urgent than others.

Further research is also needed to investigate whether inserting breaks between surgeries in the schedule can increase the stability of the schedule. The simplest way to deal with disturbances caused by uncertain surgery durations is to insert a break in the middle of the day. In this way, surgeries scheduled after the midday break can mostly start on time, even if surgeries earlier that day have taken longer than expected. Furthermore, during a break, emergency surgeries can start immediately. Therefore, it is preferable to spread the breaks of the different ORs a bit over the day, such that the overlap in breaks of ORs is small. Another approach can be to plan small breaks at the end of a surgery which duration has a high variance. Additional modeling of the BIM problem and simulation experiments should clarify the effect of BIM optimization when these considerations are taken into account.

Acknowledgements

We thank Marieke van der Lans for providing the initial ideas for this paper during her MSc thesis project at the Erasmus Medical Centre, Rotterdam.

This research is supported by the Dutch Technology Foundation STW, applied science division of NWO and the Technology Program of the Ministry of Economic Affairs.

References

[1] A. Macario, T. S. Vitez, B. Dunn, T. McDonald, Where are costs in perioperative? analysis of hospital costs and charges for inpatient surgical care, Anesthesiology 83 (1995) 1138–1144.

[2] R. L. Jackson, The business of surgery, Health Management Technology 23 (2002) 20–22.

[3] OECD, Oecd health data 2005 - statistics and indicators for 30 countries, 2005.

[4] B. Cardoen, E. Demeulemeester, J. Beliën, Operating room planning and scheduling: A literature review, European Journal of Operational Research (2010).

[5] P. J. H. Hulshof, R. J. Boucherie, J. T. van Essen, E. W. Hans, J. L. Hurink, N. Kortbeek, N. Litvak, P. T. Vanberkel, E. van der Veen, B. Veltman, I. M. H. Vliegen, M. E. Zonderland, Orchestra: an online reference database of or/ms literature in health care, Health Care Management Science 14 (2011) 383–384.

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

[7] M. Lamiri, X. Xie, A. Dolgui, F. Grimaud, A stochastic model for operating room planning with elective and emergency demand for surgery, European Journal of Operational Research 185 (2008) 1026–1037.

[8] Y. Gerchak, D. Gupta, M. Henig, Reservation planning for elective surgery under uncertain demand for emergency surgery, Management Science 42 (1996) 321–334.

[9] F. C. R. Spieksma, G. J. Woeginger, Z. Yu, Scheduling with safety distances, Annals of Operations Research 57 (1995) 251–264.

[10] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman & Co Ltd, 1979.

[11] A. Oversberg, Minimizing the waiting time for emergency surgery, Master's thesis, Universität Bonn, 2010.

[12] J. Dréo, A. Pétrowski, P. Siarry, E. Taillard, Metaheuristics for Hard Optimization, Springer, 2006.

[13] E. Aarts, J. Korst, Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing, Johh Wiley & Sons, Inc., 1989.

[14] J. Adams, E. Balas, D. Zawack, The shifting bottleneck procedure for job shop scheduling, Management Science 34 (1988) 391–401.

[15] F. Busetti, Simulated annealing overview, 2003.

[16] W. Hartholt, Beslissingsondersteuning voor het aanpassen van de online OK-planning, Master's thesis, Universiteit Twente, 2010.