Minimizing the overlap of cleaning times after surgeries while considering multiple cleaning teams

Thomas Middelkamp
Applied Mathematics, University of Twente

Supervisor: Maarten Otten, MSc *University of Twente*

Abstract

Scheduling surgeries in their preassigned Operating Room (OR) is an important task of any hospital. This paper investigates the issue of determining the order in which surgeries are being scheduled in each OR in order to minimize the cleaning time overlap after surgeries. In this way, the delay caused by a lack of cleaning teams is minimized and utilities are more efficiently used. An Integer Linear Program (ILP) is given for small instances. Several heuristic methods are introduced that are tested in a simulation study. For the simulation, real data from a Dutch hospital is used. Several realistic constraints are taken into account in the simulation. Different types of surgeries and subcategories of these surgeries are determined from the data. To test the heuristics we simulate several weeks that are reproduced from the data. The results show that the 'Prevent overlap' heuristic with four cleaning teams can reduce the average sum of the cleaning time overlap with more than 10% when compared to the original schedule.
Keywords— Scheduling, Simulation, Sequencing, Cleaning time, Heuristics, Uncertain surgery duration, Heuristics, Operating rooms

1 Introduction

Scheduling elective surgeries in Operating Rooms (OR) can be a difficult task. In the first decade of 21st century alone, [Cardoen et al., 2010] showed that over 100 papers have been published concerning operating room planning and scheduling. There are numerous factors that complicate this task, such as the scarcity of resources, conflicting priorities of stakeholders, and the variability of the length of surgeries.
Many solutions techniques are proposed by [Cardoen et al., 2010]. In order to maintain good quality of healthcare, emergency patients should be scheduled for surgery as soon as possible. However, not all ORs are suited for emergency patients, as they may lack certain equipment.
The objective to construct a schedule to minimize the waiting time of emergency patients is not uncommon. [Lamiri et al., 2008] include a column generating approach which means they consider a subset of all possible plans. This is due to the fact that the set of all possible plans is too big and thus not computationally feasible. Here, a plan is roughly an allocation of elective patients over a set of ORs in a time horizon. The costs of all surgeries in the plan are minimized in order to find the optimal schedule. These plans are optimized with a combination of heuristic methods.
It is shown by [Denton et al., 2007] that the sequence of surgeries in a particular OR can improve utilization and delays can be minimized. This is done with a simple heuristic based on which the solution is optimized with local search methods. Real data from a hospital is used and by scheduling surgeries with a higher variance later in the schedule, most goals can be attained.
When there is no suitable OR available, a patient has to wait until any OR finishes surgery. These moments when a patient can break into the schedule are called Break In Moments (BIMs). The notion of a BIM is first dicussed by [van Essen et al., 2012]. The time between two consecutive BIMs is called a Break In Interval (BII).
[van Essen et al., 2012] propose an Integer Linear Programm (ILP), local search methods and several heuristic solution methods in order to minimize the length of the maximum BII. Unfortunately, for large instances the ILP can not be solved

within a reasonable time.
[van Essen et al., 2012] showed that with the 'Fixed Goal Values' heuristic, the average maximum BII can be reduced with 10% when compared to a schedule of the Erasmus Medical Centre. Numerous other solution methods are proposed. In all of the solution methods mentioned in this paper, it is assumed that the surgery times, at the times of scheduling, are deterministically known. However, surgery times are far from deterministic due to complications and other unforeseen events. Therefore [Vandenberghe et al., 2019] suggested the notion of a Stochastic Break In Moment (SBIM), where surgery durations are modeled as independent random variables. In their paper they introduce various methods on how to solve this problem. The methods include a Mixed Interger Linear Programm (MILP) using Sample Average Approximation (SAA), various local search methods and a variance based heuristic.

The papers mentioned previously do not consider the cleaning times after each surgery. If there are no cleaning teams available, then overlapping cleaning times of different ORs will result in an increase in the waiting time of to be performed surgeries. In order for all planned surgeries to take place on the preassigned date, it is desired to construct a schedule in such a way that the overlap of completion times is minimized.

The objective of our problem is substantially different than the objective of the (S)BIM problem. In the (S)BIM problem, the objective is to minimize the length of a the maximum BII, while our target is to minimize the overlapping cleaning times. However, the (S)BIM problem is of great usage, its framework allows us to adapt and construct heuristics in order to minimize the overlapping cleaning times.
Minimizing the cleaning overlap can be seen as a Scheduling with Safety Distance (SSD) problem. The goal of the SSD problem is to schedule jobs in such a way that their completion time is a certain distance d apart. This is discussed by [Spieksma et al., 1995]. In the BIM problem, SSD translates to maximizing the minimum BII. Even though the methods used in the (S)BIM and the SSD problems are different, the outcome of the solution methods of the (S)BIM and SSD problem is to spread the BIMs as evenly as possible over the day.
The main goal of this paper is to test heuristics by means of a realistic simulation model. Various heuristics and an ILP are introduced. Notation introduced by [van Essen et al., 2012] will be used in the ILP. In the model, real data is used from a dutch hospital and realistic scheduling constraints are taken into account. One of the constraints is to take the working schedule of surgeons into account. This is done according to a block structure. The block structure is introduced to more realistically In an OR, consecutive surgeries can only be scheduled if they are of the same type as the previous surgery. When no surgeries of the same type have yet to be scheduled, a new type of surgery is allowed to be scheduled, creating a new block. The other constraints are to schedule children as early as possible in the day and infectious patients at the end of the day.
The remainder of this paper is structured as follows. In section 2 we define the BIM problem and develop it to match our different objective. In section 3, an ILP and several constructive heuristics are presented. In section 4 it is explained how several parameters that are used in the simulation are extracted from the data. In section 5, assumptions and results of the simulation are discussed. In section 6, we summarize this paper and look into further research of this topic.

2 Problem description

Before defining the problem, let us first visualize how the BIMs and BIIs look like in a schedule. For simplification, we will use three ORs.

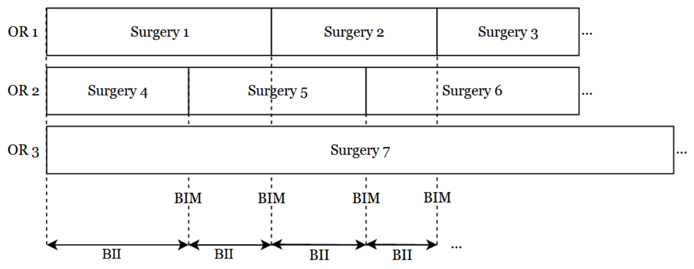

Figure 1: Break in Moments and Break In Intervals.

As can be seen, the BIMs occur whenever a surgery ends, the interval between two consecutive BIM is the BII.
However, Figure 1 is not accurate with the situation in reality. Every OR is cleaned after surgery. A delay in one or more ORs occurs in the following situation. If more ORs need to be cleaned at the same time but there are less cleaning teams than ORs. This will cause delay in one or more ORs. Therefore, we want to prevent that the cleaning times of ORs overlap, as this decreases the probability of causing a delay.
The delay is indicated in Figure 2.

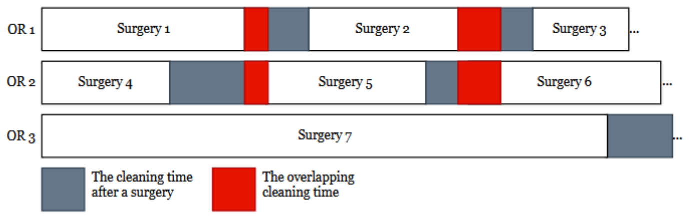

Figure 2: The schedule with the overlapping cleaning time indicated.

The delays have an effect on the rest of the schedule that day. As can be seen in Figure 2, the delay in OR 1 causes a delay of the end time of surgery 3. It might occur that the cleaning time of surgery 3 overlaps with a cleaning time of another OR, while this was not planned in the original schedule. This overlap later in the day is caused by an overlap earlier that day. The effects of the delay in this example are displayed in Figure 3.

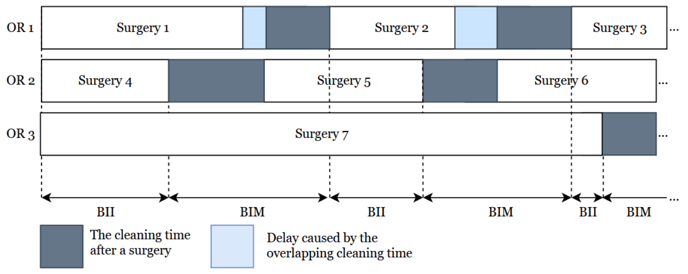

Figure 3: The adjusted schedule.

An emergency patient can be treated only if an OR is available or during any moment of when cleaning is done. Because of this, instead of the BIMs being moments, the BIMs become intervals. In the worst case, an emergency patient arrives just after the start of the largest BII. Therefore, in the BIM problem, it suffices to schedule the elective surgeries such that the maximum BII is minimized.

Let $I = \{1, 2, \ldots, M\}$ be the set of surgeries. Each surgery $i \in I$ is of a certain type, e.g. Ear Nose and Throat (ENT), gynaecology (GYN) or Urology (URO). The type of surgery i is denoted by $t \in T$, where $T = \{1, 2, \ldots, S\}$ is the set of all distinct types of surgeries. We write i_t if surgery $i \in I$ is of type $t \in T$. We will omit this notation if the type of surgery is irrelevant in the context. The duration of the surgery $i_t \in I, t \in T$ is denoted as p_{i_t}. The set of ORs is given by $J = \{1, 2, \ldots, N\}$. The set of surgeries assigned to OR $j \in J$ is $I_j \subset I$. Let $O(i)$ denote the the OR in which surgery $i \in I$ can be performed. We define a block B to be a set of surgeries of the same type. Thus the set of surgeries in OR $j \in J$, $I_j = \{B_{t_1}, B_{t_2}, \ldots, B_{t_m}\}$, where $t_r \in T$ for $1 \leq r \leq S$. After every surgery, the OR in which the surgery was performed has to be cleaned. For surgery $i_t \in I, t \in T$ denote the cleaning time of this surgery as L_{i_t}.

The interval of which surgeries are performed in all ORs is called the occupied interval. The time at which OR $j \in J$ begins its first surgery is defined as s_j, the time at which OR $j \in J$ finishes its last surgery is defined as e_j.

The interval starts at time S, when the first OR starts performing surgeries, and ends at time E, when the last OR completes all of its surgeries. In contrast with [van Essen et al., 2012] and [Vandenberghe et al., 2019] we define the occupied interval as the time that any OR is active. This is needed as the objective is different. The (S)BIM problem is concerned with minimizing the time until the next BIM occurs. We are interested in the minimum overlap of cleaning times, so even if an OR is available the cleaning times should still be minimized.

The OR scheduling problem can be seen as scheduling jobs on machines. Here the ORs are viewed as 'machines' and the surgeries that have to be scheduled as 'jobs'.

3 Solution methods

In this section we discuss two types of solution methods. We will first introduce an Integer Linear Programm (ILP) in subsection 3.1. In subsection 3.2 several constructive heuristics are proposed. The solution methods are adapted from the paper by [Amalia, 2018], this paper is a bachelor thesis from the University of Twente.

3.1 Exact solution method

We will use the framework of the ILP of [van Essen et al., 2012], however we do have to modify certain constraints such that the ILP fits our context. These modifications are introduced in [Amalia, 2018].

We assume the duration of each surgery and cleaning time is deterministic. However, these times depend on the surgery. Assume for this solution method that there is only one cleaning team. In order to solve this scheduling problem, we will have to determine all the BIM of all ORs. To be able to determine the BIMs, we define a local and a global sequence. Consider one specific OR, the consecutive completion times C_i of surgeries in this OR is called the local sequence. The global sequence is defined as the consecutive completion times C_i over all ORs.

We define the global BIM sequence to be all the BIMs of all the ORs. The local BIM sequence is a sequence of all the BIMs of one OR. In this sequence, the BIMs are sorted in increasing order. From this global sequence, we can determine the lengths of the BII. An example of a global sequence is shown in Figure 4.

The global sequence is defined by its BIMs. Note that the BIMs

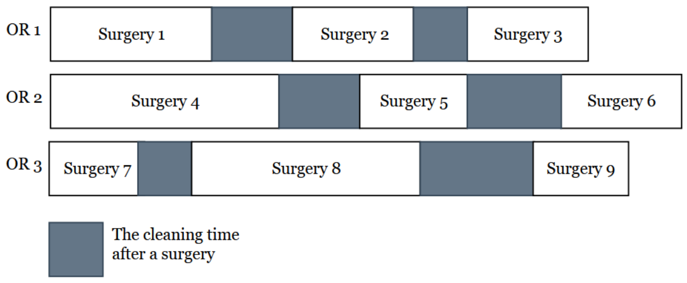

Figure 4: Global sequence: (7,1,4,2,8,5,3,9,6)

itself are defined by completion and cleaning times of the surg-

eries. The completion time of surgery $i \in I$ will be denoted by C_i. We introduce a binary variable that defines the global sequence. Let Y_{ik} take on value 1 whenever surgery $i \in I$ is before surgery $k \in I$ in the global sequence. If surgery k is scheduled before surgery i in the global sequence, we have $Y_{ki} = 1$.
In order that only one of the two Y_{ik} and Y_{ki} is 1, we need the following constraint:

$$Y_{ik} + Y_{ki} = 1, \forall i, k \in I, i \neq k. \tag{1}$$

Let the global position of surgery $i \in I$ be denoted by Z_i. We obtain an expression for Z_i in terms of Y_{ki}.

$$Z_i = 1 + \sum_{i \in I} Y_{ki}, \forall i, k \in I, i \neq k. \tag{2}$$

Note that if $Y_{ik} = 1$, that means that surgery i is scheduled before surgery k, thus $Z_i < Z_k$. Thus we need the following constraint:

$$Z_i \leq Z_k + \mathcal{M} Y_{ki}, \forall i, k \in I, i \neq k. \tag{3}$$

Note that if $Y_{ik} = 1$, the constraint is directly satisfied. On the other hand, if $Y_{ki} = 1$, then we can find a sufficiently large number $\mathcal{M}$ such that $Z_i \leq Z_k$.
The local sequence has to be consistent with the global sequence. Recall that if surgery $i \in I$ is before surgery $j \in I$ in the local sequence, this means that the completion time of surgery $i \in I$ is less than the completion time of surgery $j \in I$.
This means that in the global sequence we must have $Y_{ik} = 1$, which is established in the following constraint:

$$C_i - \mathcal{M}(1 - Y_{ik}) = C_k, \forall i, k \in I, i \neq k. \tag{4}$$

A closed expression for the completion time, C_i of surgery $i \in I$ can be obtained as follows. Start by adding the duration, including cleaning times, of all surgeries that occurred in the OR where i will take place, say OR j. Furthermore, add the time when the OR performed its first surgery, s_j, and the duration of surgery $i \in I$, p_i.

$$C_i = \sum_{k \in I_{O(i)}} (p_k + L_k) Y_{ki} + p_i, \forall i, k \in I, i \neq k. \tag{5}$$

Let X_{ik} be the amount of time the cleaning windows of surgery $i, k \in I$ overlap. For most surgeries, the overlap of cleaning windows will be zero as they are scheduled too far apart.
However, if $Y_{ik} = 1$ we want to make sure that the cleaning windows do not overlap. Thus the completion time of surgery $i \in I$ plus the cleaning time of surgery $i \in I$ should be less than or equal to the completion time of surgery $k \in I$ plus the overlap X_{ik}. We add X_{ik} because during this time, the cleaning team is cleaning OR $I_{O(i)}$, so overlap in cleaning windows can not happen. This is given in the following constraint:

$$C_i + L_i \leq C_k + X_{ik} + \mathcal{M}(1 - Y_{ik}), \forall i, k \in I, i \neq k. \tag{6}$$

Our overall goal is to minimize X_{ik} $\forall i, k \in I, i \neq k$. Thus we obtain the following minimization problem:

$$\begin{aligned} min \quad & \sum_{\substack{i,k \in I \\ i \neq k}} X_{ik}, & (7) \\ s.t. \quad & (1) - (6), \\ & Y_{ik} \in \{0, 1\};\ X_{ik} \in \mathbb{R}\ \forall i, k \in I,\ i \neq k, \\ & C_i, L_i, p_i \in \mathbb{R}^+; Z_i \in \mathbb{N}\ \forall i \in I. \end{aligned}$$

It has been shown by [Amalia, 2018] that solving this ILP is NP hard . Only solutions for small instances can be obtained within a reasonable time. The goal of this paper is to construct a schedule in the realistic case. That is why we choose to not solve this ILP. In subsection 3.2 faster approximation methods are introduced.

3.2 Constructive heuristics

In this section, we will adapt the heuristic methods presented in [Amalia, 2018]. The heuristics need to be changed in order to fit the realistic scheduling constraints. In order to keep the Algorithms 1-5 readable, only the block structure will be added. A schedule will be determined with each heuristic. A simulation study will be done in order to test each heuristic against various situations. Each heuristic is adapted according to a block structure. Each OR is preassigned with blocks of surgeries, each OR has a 'current' block, the first nonempty block in the local sequence. Due to the this block structure we do not have the freedom of choosing every surgery. We denote the current block as $B_c \in I_j$. For OR $j \in J$ we can only select a surgery i from our current block B_c.

3.2.1 Midpoint fixed goals with blocks

In this method, we try to schedule a surgery one by one such that the duration added by half of the cleaning time is as close to a target value λ as possible. For example, consider a situation with four ORs where three surgeries are assigned to each OR. Suppose the occupied interval is twelve hours. We would like the completion time added by half of the cleaning time after each surgery takes place at $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, if we assume the occupied interval starts at time zero.
What we try to accomplish with this heuristic is that the BIM's are spread as evenly as possible over the occupied interval. The target value λ can be calculated as follows. Recall that S and E are the start and end time of the occupied interval, respectively. Here M_j denotes the number of assigned surgeries in OR $j \in J$.

$$\lambda = \frac{E - S}{\sum_{j \in J} M_j} = \frac{E - S}{M}. \tag{8}$$

Algorithm 1: Fixed goals midpoint with blocks

```
Determine $\lambda$, set $t = 1, I' = \emptyset$, $S_j = s_j$ for all $j \in J$ and
  $done = False$;
while not done do
    for each OR j in 1,2,...,N do
        Let $B_c$ be the current block of surgeries;
        Determine for all the unscheduled surgeries in that
          block the completion time $C_i = S_{O(i)} + p_i$;
    end
    Set $s = \min_{j \in J} S_j$;
    Determine what surgery has to be scheduled next.
      $k = \arg\min_{i \in B} |s + \lambda t - (C_i + \frac{1}{2} L_i)|$;
    For OR O(k), update the start time
      $S_{O(k)} = S_{O(k)} + (p_k + L_k)$;
    Delete surgery $k$ from I, i.e. $I = I \setminus \{k\}$;
    Add surgery $k$ to our schedule $I'$, i.e. $I' = I' \cup \{k\}$;
    Set $t = t + 1$;
    if I is empty then
        $done = True$;
    end
end
```

3.2.2 Endpoint fixed goals with blocks

This heuristic is very similar the the heuristic proposed in Section 3.2.1. However, now we try to schedule the surgeries such that the completion time added by the cleaning time is as close to λ in (8) as possible. This is illustrated in algorithm 2.

Algorithm 2: Fixed goals endpoint with blocks

```
Determine $\lambda$, set $t = 1, I' = \emptyset$, $S_j = s_j$ for all $j \in J$ and
  $done = False$;
while not done do
    for each OR j in 1,2,...,N do
        Let $B_c$ be the current block of surgeries;
        Determine for all the unscheduled surgeries in that
          block the completion time $C_i = S_{O(i)} + p_i$;
    end
    Set $s = \min_{j \in J} S_j$;
    Determine what surgery has to be scheduled next.
      $k = \arg\min_{i \in B} |s + \lambda t - (C_i + L_i)|$;
    For OR O(k), update the start time
      $S_{O(k)} = S_{O(k)} + (p_k + L_k)$;
    Delete surgery $k$ from I, i.e. $I = I \setminus \{k\}$;
    Add surgery $k$ to our schedule $I'$, i.e. $I' = I' \cup \{k\}$;
    Set $t = t + 1$;
    if I is empty then
        $done = True$;
    end
end
```

3.2.3 End point flexible goals with blocks

In the proposed heuristics, the goal value λ in (8) is calculated once before the algorithm schedules the surgeries. Note that the value λ depends on the total number of surgeries. So when there are surgeries scheduled, this values changes. That is why in this heuristic, we update the value of λ after each time a surgery is scheduled. In this way, we get a more accurate value of λ when certain surgeries have already been scheduled.
There are two things that change in each iteration that is needed to calculate λ, the remaining occupied interval and the number of to schedule surgeries.
Denote P as the start of the remaining occupied interval. Let K denote the number of scheduled surgeries. Now λ is defined as follows.

$$\lambda = \frac{E - P}{\sum_{j \in J} M_j - K} = \frac{E - P}{M - K}. \tag{9}$$

The implementation can be seen in algorithm 3

Algorithm 3: Flexible goals endpoint with blocks

```
Set $t = 1, I' = \emptyset$, $S_j = s_j$ for all $j \in J$, $K = 0$ and
  $done = False$;
while not done do
    Set $P = \min_{j \in J} s_j$, determine $\lambda$;
    for each ORs j in 1,2,...,N do
        Let $B_c$ be the current block of surgeries;
        Determine for all the unscheduled surgeries in that
          block the completion time $C_i = S_{O(i)} + p_i$;
    end
    Set $s = \min_{j \in J} S_j$;
    Determine what surgery has to be scheduled next.
      $k = \arg\min_{i \in B} |s + \lambda t - (C_i + L_i)|$;
    For OR O(k), update the start time
      $S_{O(k)} = S_{O(k)} + (p_k + L_k)$;
    Delete surgery $k$ from I, i.e. $I = I \setminus \{k\}$;
    Add surgery $k$ to our schedule $I'$, i.e. $I' = I' \cup \{k\}$;
    Set $t = t + 1$;
    Set $K = K + 1$;
    if I is empty then
        $done = True$;
    end
end
```

3.2.4 Prevent overlap

We start with selecting the surgery with the shortest completion and cleaning time all together. We call this surgery $k \in I$. In this way, the idle time of the cleaning team in the beginning of the day is minimized. Doing this enables us to increase coverage of the cleaning times over the occupied interval. This facilitates spreading the cleaning intervals more evenly over the day.

The objective is to find another surgery $w \in I$ such that there is no overlapping cleaning time.
Consider the end of the cleaning time of surgery k. In the most ideal situation, surgery $w \in I$ finishes as close as possible to the end of the cleaning time of surgery $k \in I$. We can find this surgery $w \in I$ systematically if we take $w = \arg\min_{i \in I}\{C_k + L_k - C_i\}$. This heuristic is illustrated in the algorithm 4.

Algorithm 4: Prevent overlap

```
Set $I' = \emptyset$, $S_j = s_j$ for all $j \in J$ and $done = False$;
while not done do
    for each OR j in 1,2,...,N do
        Let $B_c$ be the current block of surgeries;
        Determine for all the unscheduled surgeries in that
          block the completion time $C_i = S_{O(i)} + p_i$;
    end
    Set k = $\arg\min_{i \in I}(C_i + L_I)$;
    Schedule surgery $k$ in OR O(k);
    For OR O(k), update the start time
      $S_{O(k)} = S_{O(k)} + (p_k + L_k)$;
    Delete surgery $k$ from I, i.e. $I = I \setminus \{k\}$;
    Add surgery $k$ to our schedule $I'$, i.e. $I' = I' \cup \{k\}$;
    for each OR j in 1,2,...,N do
        Let $B_c$ be the current block of surgeries;
        Determine for all the unscheduled surgeries in that
          block the completion time $C_i = S_{O(i)} + p_i$;
    end
    Set $w = \arg\min_{i \in I}\{C_k + L_k - C_i\}$;
    Schedule surgery $w$ in OR O(w);
    For OR O(w), update the start time
      $S_{O(w)} = S_{O(w)} + (p_w + L_w)$;
    Delete surgery $w$ from I, i.e. $I = I \setminus \{w\}$;
    Add surgery $w$ to our schedule $I'$, i.e. $I' = I' \cup \{w\}$;
    if I is empty then
        done is True;
    end
end
```

3.2.5 One by one sequencing

Similarly to heuristic 'Prevent overlap' we start by selecting the surgery with the shortest total time. Again, we call this surgery $k \in I$. In choosing the next surgery $i \in I$, we try to not schedule surgeries that will result in overlap of cleaning times. Choosing surgery $i \in I$ can be chosen from two perspectives. One perspective is that the completion time added by the cleaning time of surgery i is as close as possible to the completion time of surgery k.
The other perspective is to choose the next surgery is to select one such that the completion time added by the cleaning time of surgery k is as close as possible to the completion time of surgery i. Thus, choosing such a surgery can be realised according to the following $w = \arg\min_i\{(C_k - (C_i + L_i))^+, (C_i - (C_k + L_k))^+\}$. This is illustrated in Figure 5. Both surgeries i_1 and i_2 do not cause an overlap in cleaning time.
Only positive values are considered. This is because if both values are negative, this minimization does not minimize the overlapping cleaning time. This is illustrated in Figure 6.
In order to minimize this overlap, we choose $w = \arg\max_{i \in I}\{(C_k - (C_i + L_i)), (C_i - (C_k + L_k))\}$. The heuristic is illustrated in the following algorithm.

Algorithm 5: One by one sequencing

```
Set $I' = \emptyset$, $S_j = s_j$ for all $j \in J$ and $done = False$;
while not done do
    for each OR j in 1,2,...,N do
        Let $B_c$ be the current block of surgeries;
        Determine for all the unscheduled surgeries in that
          block the completion time $C_i = S_{O(i)} + p_i$;
    end
    Set k = $\arg\min_{i \in I}(C_i + L_I)$;
    Schedule surgery $k$ in OR O(k);
    For OR O(k), update the start time
      $S_{O(k)} = S_{O(k)} + (p_k + L_k)$;
    Delete surgery $k$ from I, i.e. $I = I \setminus \{k\}$;
    Add surgery $k$ to our schedule $I'$, i.e. $I' = I' \cup \{k\}$;
    for each OR j in 1,2,...,N do
        Let $B_c$ be the current block of surgeries;
        Determine for all the unscheduled surgeries in that
          block the completion time $C_i = S_{O(i)} + p_i$;
    end
    if $C_k - (C_i + L_i) > 0$ or $C_i - (C_k + L_k) > 0$ then
        $w = \arg\min_i\{(C_k - (C_i + L_i))^+, (C_i - (C_k + L_k))^+\}$;
    end
    if for all $i \in B_c$ $C_k - (C_i + L_i) < 0$ and
      $C_i - (C_k + L_k) < 0$ then
        $w = \arg\max i\{(C_k - (C_i + L_i)), (C_i - (C_k + L_k))\}$;
    end
    Schedule surgery $w$ in OR O(w);
    For OR O(w), update the start time
      $S_{O(w)} = S_{O(w)} + (p_w + L_w)$;
    Delete surgery $w$ from I, i.e. $I = I \setminus \{w\}$;
    Add surgery $w$ to our schedule $I'$, i.e. $I' = I' \cup \{w\}$;
    if I is empty then
        done is True;
    end
end
```

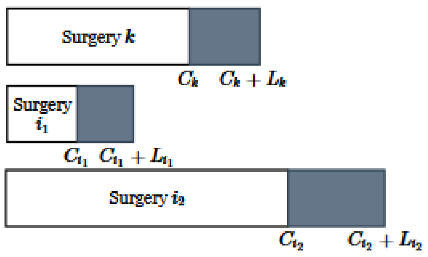

Figure 5: Either one of these scenarios is preferred.

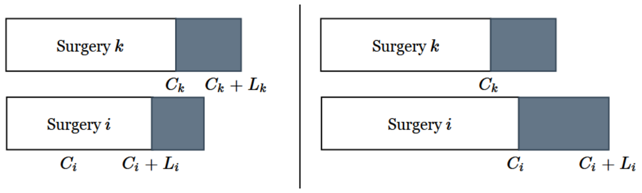

Figure 6: A situation when the cleaning times will always overlap.

4 Data analysis

A table with all the extracted data that is used in the simulation can be found in Appendix A. For the simulation, real data from the 'Jeroen Bosch Ziekenhuis' (JBZ) from July 2016 until July 2018 is used. The JBZ has 16 ORs with 20 different surgery types that can be performed. On average, 54 surgeries take place each day. In the proposed heuristics in section 3.2, it is assumed that for each surgery $i_t \in I$ of type $t \in T$, the duration p_{i_t} is deterministically known. Using the heuristics, a schedule is obtained that is simulated. For the simulation, we assume that each surgery $i_t \in I$ of type $t \in T$ has a three parameter lognormal distribution with $E[p_{i_t}] = \mu_{i_t}$ and Var $= \sigma^2_{i_t}$. [Strum et al., 2000] suggest that surgery durations are most realistically modelled with a lognormal distribution, and especially the three parameter lognormal distribution, which has been shown by [Spangler et al., 2004] and [Stepaniak et al., 2010]. The third parameter which we will call γ is a shift parameter. The parameters are determined using the scipy.stats module in Python. This module allows for negative values of γ, which is unconventional according to [Singh, 1998]. The data consists of over 30.000 surgeries, the type of each surgery is indicated. All types $t \in T$ can be characterized in this way. In order to obtain a realistic distribution, surgery types are not considered if there are less than one hundred occurrences in our data set. Each surgery has an expected duration. On the expected durations, a lognormal fit is obtained using Python. In Figure 7, an example is shown for anaplasty type surgeries.
The height of each peak represents the percentage of surgeries that have this planned duration.
As can be seen in Figure 7, the lognormal fit over the entire data set of planned durations is not satisfactory. This means that drawing random samples from the distribution is not representative for

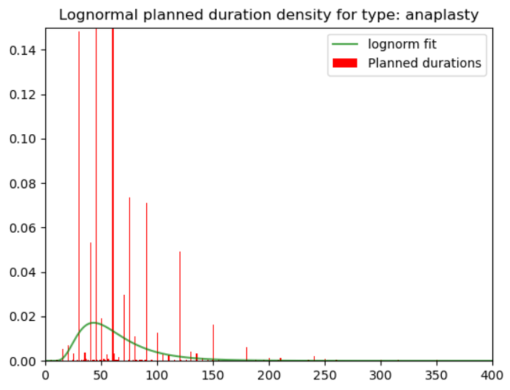

Figure 7: An example of a lognormal fit on the surgery type anaplasty. The density on the amount of surgeries is considered.

all the surgeries. Therefore, subcategories of the surgeries are introduced. A subcategory is chosen as a duration that has more than 5% occurrences in the total amount of surgeries of that type. The rest of the surgeries are added to the closest duration peak higher than 5%. The merged peaks for anaplasty type surgeries is shown in Figure 8. The original durations are shifted slightly to the left to prevent overlap, in order to clarify the figure.

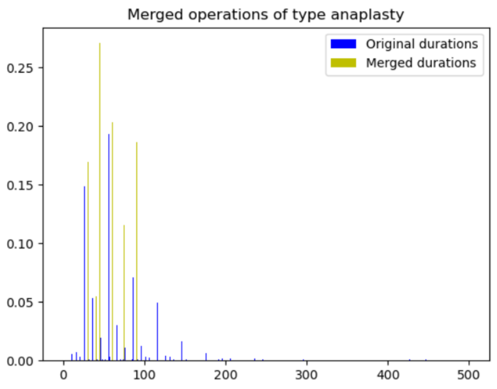

Figure 8: The grouped surgeries to all the peaks that are higher than 5%.

For each of these subcategories, a lognormal fit is made based on the real duration. An example is shown in Figure 9 and Figure 10. This is done for each type of surgery in order for the simulation to be more realistic.
The cleaning time that is scheduled by the JBZ is always 15 minutes. The cleaning time after a surgery depends on its type and duration. The mean of all durations L_{i_t} for all $i \in I$, $t \in T$ are obtained from the data and used in the simulation. It is assumed that the cleaning times in the simulation are deterministic.
In determining the schedule, three realistic scheduling constraints are taken into account. The first is the block schedule to more realistically display the schedule of the surgeons. The second is

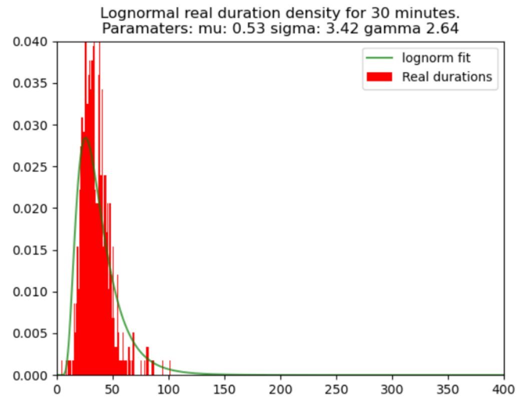

Figure 9: A lognormal fit for anaplasty type surgeries of the 30 minute subcategory.

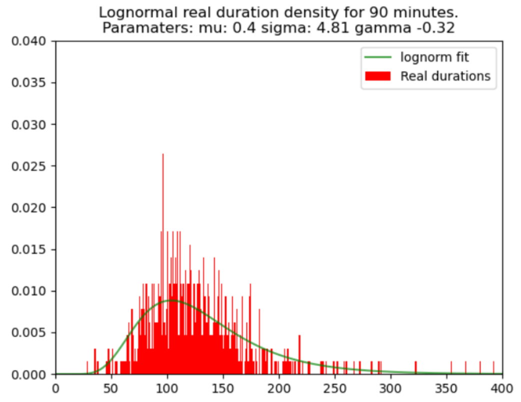

Figure 10: A lognormal fit for anaplasty type surgeries of the 90 minute subcategory.

that children should be scheduled as early as possible. There is a possibility that more than one type of surgery is assigned to an OR, and at least one surgery of each type concerns a child. In this case, the block structure is preserved and there is a possibility that a child is scheduled later on the day. Lastly, surgeries that concerns patients with an infectious disease should be scheduled last in a day to prevent contamination.
The proportion of surgeries $i_t \in I$, $t \in T$ which are children or infectious are obtained from the data and used in the simulation.
The schedule of any given day can be reconstructed. A schedule consists of a set of ORs that correspond to a set of surgeries that have been scheduled in that OR. From these surgeries, the start and end times are known. Using this information, the lengths of the BIIs can be calculated. A BII is assumed to take place in between two surgeries. Thus ORs that are inactive are not taken into account for the calculation of lengths of the BIIs.

5 Simulation

Two randomly selected weeks are selected for the simulation. This is done in order to create a view on how the heuristics perform in a realistic situation. The average number of surgeries per day is 54. To maintain a realistic view, we choose that there should be at least 400 surgeries performed in that week. Weeks that contain days that have 0 surgeries are omitted.
A new order of the surgeries of each OR are determined using the heuristics of Section 3.2. In determining the order of the surgeries, it is assumed that for all $i_t \in I$, $t \in T$, the planned durations p_{i_t} and the planned cleaning times L_{i_t} are deterministic. Because the lognormal distribution is a skewed distribution, the planned duration of a surgery $i_t \in I$, $t \in T$ is the median of the three parameter lognormal distribution. The planned cleaning times are chosen as follows, if $p_i < 30$ then $L_i = 5$, otherwise $L_i = 15$. After the schedule is determined by a heuristic method, it is simulated. The number of cleaning teams can be set as a variable in the simulation. Consider Figure 3, if there were two cleaning teams present, there would not have been any overlap of cleaning times. From the figure, it is clear that the increase of cleaning teams can have a negative effect on minimizing the length of the maximum BII. This is because if ORs are waiting for cleaning, emergency surgery can take place. Cases with three and four cleaning teams are considered only, as the results using less cleaning teams are insufficient. Each evaluated schedule is simulated 500 times. Confidence intervals (95%) are provided for the average duration of an overlap and the sum of all overlaps. An overlap is accounted for only if there is a delay caused by it. The results of two randomly selected weeks are presented in Table 1 and Table 2. In simulating the real schedule and the schedules determined by the heuristics, we assume that there is no idle time in between surgeries or idle time at the start of a day, even though if this was the case on that day.
The simulation model is implemented in Python 3.7.
The confidence intervals of the average overlap of two cleaning intervals all tend to converge to the same values. Even though the schedules are different, the set of surgeries that is being scheduled in an OR is the same for all heuristics. It is assumed that cleaning times are deterministic in the simulation. Therefore, using different heuristics to construct the schedule, the same amount of cleaning time takes place in each OR. Apparently, in the simulation, the average amount of overlap between any two cleaning intervals throughout the day that cause a delay approaches a constant.
The results show that only the Prevent overlap (PV) and the One by one sequencing (OOS) heuristic always improve on the original schedule. However, PV performs better in all instances than OOS.
Both of these heuristics schedule two surgeries in each iteration. The surgery with the shortest total time is chosen, a second surgery is selected to prevent or minimize the overlap. The

Table 1: Instance of a random selected week. There were 405 surgeries this week. Conf. int. avg. overlap (min) = average overlap of two cleaning windows in minutes;Conf. int. sum overlap (min) = the total overlap in minutes;Conf. int.max. BII (min) = the longest length of a Break In Intervals in minutes.

Solution methods	Runtime (s)	Conf. int. avg. overlap (min)	Conf. int. sum overlap (min)	Conf. int. max. BII (min)
		Three cleaning teams		
Original schedule	70.82	2.91 ±4.2	61.31 ±46.08	48.98 ±91.05
Midpoint fixed goals	72.99	2.93 ±4.22	63.97 ±49.88	46.44 ±78.5
Endpoint fixed goals	73.1	2.92 ±4.21	64.72 ±48.32	48.79 ±102.77
midpoint flexible goals	71.63	2.96 ±4.25	68.89 ±56.16	51.46 ±93.98
Prevent overlap	71.2	2.83 ±4.12	55.07 ±42.29	47.35 ±79.54
One by one sequencing	73.19	2.84 ±4.12	57.34 ±41.29	49.93 ±118.39
		Four cleaning teams		
Original schedule	30.92	2.12 ±3.28	14.84 ±19.07	47.79 ±87.19
Midpoint fixed goals	33.16	2.14 ±3.29	17.1 ±22.83	46.59 ±75.73
Endpoint fixed goals	33.61	2.13 ±3.26	16.04 ±20.26	45.59 ±79.46
midpoint flexible goals	32.86	2.17 ±3.29	19.52 ±25.94	50.64 ±91.81
Prevent overlap	32.58	2.07 ±3.21	12.75 ±16.93	50.71 ±111.63
One by one sequencing	32.55	2.07 ±3.21	12.83 ±17.37	49.68 ±119.7

Table 2: Instance of a random selected week. There were 439 surgeries this week. Conf. int. avg. overlap (min) = average overlap of two cleaning windows in minutes;Conf. int. sum overlap (min) = the total overlap in minutes;Conf. int.max. BII (min) = the longest length of a Break In Intervals in minutes.

Solution methods	Runtime (s)	Conf. int. avg. overlap (min)	Conf. int. sum overlap (min)	Conf. int. max. BII (min)
		Three cleaning teams		
Original schedule	59.0	3.03 ±4.41	61.5 ±49.74	64.18 ±100.92
Midpoint fixed goals	75.89	3.02 ±4.43	58.67 ±44.77	59.61 ±91.26
Endpoint fixed goals	91.78	3.0 ±4.38	59.67 ±46.3	60.89 ±93.15
midpoint flexible goals	75.16	3.01 ±4.4	64.72 ±48.64	59.8 ±101.87
Prevent overlap	66.81	2.99 ±4.36	57.16 ±46.03	57.27 ±87.99
One by one sequencing	53.11	3.02 ±4.38	58.81 ±46.36	57.37 ±86.05
		Four cleaning teams		
Original schedule	34.71	2.28 ±3.54	14.57 ±19.41	63.63 ±93.71
Midpoint fixed goals	88.09	2.25 ±3.54	13.92 ±18.07	61.31 ±92.03
Endpoint fixed goals	68.09	2.25 ±3.54	14.34 ±18.49	60.55 ±100.81
midpoint flexible goals	35.71	2.26 ±3.54	15.23 ±19.38	58.96 ±83.64
Prevent overlap	34.02	2.24 ±3.52	13.3 ±17.93	56.49 ±89.31
One by one sequencing	34.38	2.22 ±3.48	13.7 ±17.94	57.13 ±84.09

other heuristics, i.e. 'Fixed' and 'Flexible goals', aim to spread the cleaning intervals as evenly as possible over the day. These frequently perform worse than the original schedule.
The reason that PV and OOS perform better could be due to the fact that there are many surgeries of all sorts of durations, as can be seen in Figure 11 and Figure 12. This can be beneficial for PV and OOS as there is more freedom in choosing the second surgery in the iteration. This can be unfavourable for the other heuristics as the target λ might not be attained later on in the day the because there are no surgeries left that are close to λ.
A reason why all heuristics might perform worse than expected is that it is assumed that all surgeries start at time $t = 0$. This results in a peak workload for the cleaning teams in the beginning of the day. The high work load results in cleaning time overlaps which in turn effects the rest of the schedule in that OR. The heuristics do not account for the already shifted schedule. Because of this The peak contributes to overlap which affects the rest of the schedule in that OR. In creating the schedule using the heuristics, the overlap and its effects are not accounted for.
Even though the objective was not to minimize the maximum BII, it is interesting to examine the performances. The variance of the conf. int. of the maximum BII is very broad, using more cleaning teams resolves this. Another factor of this is the high variance of the surgery durations. The purpose of the fixed and flexible goals heuristic is to spread the cleaning intervals as evenly as possible over the days. This shows in Table 1, the maximum BII is the lowest amongst the other heuristics. In Table 2, PV and OOS perform better on minimizing the maximum BII.
We conclude that the PV heuristic algorithm performs the best. The average sum of the overlap is the lowest amongst all other

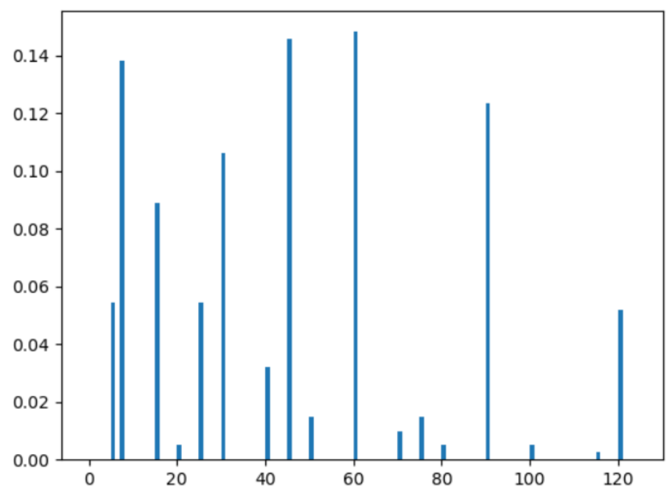

Figure 11: The duration density of the planned durations of the week in Table 1

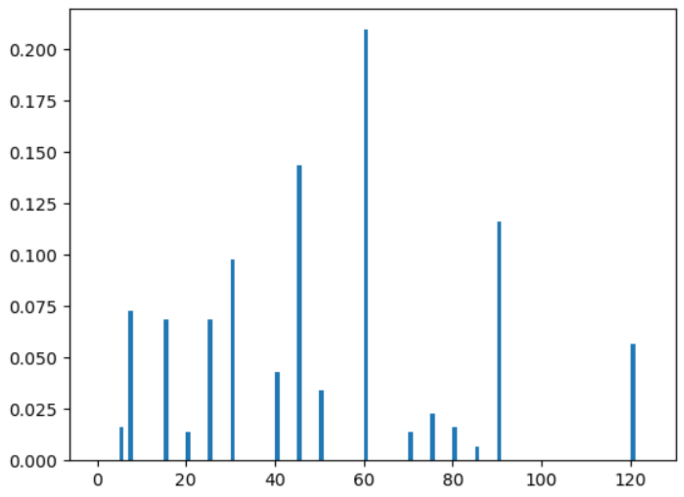

Figure 12: The duration density of the planned durations the week in Table 2

heuristics. Furthermore, the average maximum BII is minimized in most cases.

6 Conclusions and recommendations

We have studied the problem to minimize overlap of cleaning times. We built on the framework provided by [Amalia, 2018]. We have used this framework and introduced an ILP and several heuristic methods. The heuristics are adjusted in order to create a more realistic scenario. A simulation study on reconstructed weeks using date from a Dutch hospital has been done to test several solution methods. In the simulation, several realistic constraints are taken into account. A block structure of consecutive surgeries of the same type is introduced. Elective surgeries that concern children should be scheduled early on the day and infectious surgeries are scheduled at the end of the day.
The 'Prevent overlap' algorithm provides the best results in minimizing the average sum of cleaning overlap and minimizing the maximum BII. The average sum of the overlap can be reduced by more than 10%, provided that four cleaning teams are in use. Therefore, we recommend hospitals to use the prevent overlap heuristic and use four cleaning teams. Both the average sum of cleaning overlap and the length of the maximum BII can be reduced substantially.
Further research can focus on whether inserting idle time can have a positive effect on minimizing the cleaning time overlap. This can be at the beginning of a day or in between surgeries. Idle time can be scheduled in ORs if it is expected that the completion times of the surgeries are close to each other. In this way, the completion times can be shifted which might result in fewer overlaps. Inserting idle can be beneficial in the results of the BIM problem too, as inserted idle time in can reduce the length of a BII.

Acknowledgement

I would like to thank Maarten Otten, Msc, for all the support and insightful conversations about the thesis. I could not have imagined a more pleasant interaction between student and supervisor. I thank ir. Jasper Bos for providing the data from the Jeroen Bosch Ziekenhuis.

References

[Amalia, 2018] Amalia, O. A. (2018). Surgery Sequencing in Multiple Operating Rooms while Considering Cleaning Time Windows. (1973177).

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

[Denton et al., 2007] Denton, B., Viapiano, J., and Vogl, A. (2007). Optimization of surgery sequencing and scheduling decisions under uncertainty. *Health Care Management Science*, 10(1):13–24.

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

[Singh, 1998] Singh, V. P. (1998). Three-Parameter Lognormal Distribution. pages 82–107. Springer, Dordrecht.

[Spangler et al., 2004] Spangler, W. E., Strum, D. P., Vargas, L. G., and May, J. H. (2004). Estimating procedure times for surgeries by determining location parameters for the lognormal model.

[Spieksma et al., 1995] Spieksma, F. C., Woeginger, G. J., and Yu, Z. (1995). Scheduling with safety distances. *Annals of Operations Research*, 57(1):251–264.

[Stepaniak et al., 2010] Stepaniak, P. S., Heij, C., and de Vries, G. (2010). Modeling and prediction of surgical procedure times. *Statistica Neerlandica*, 64(1):1–18.

[Strum et al., 2000] Strum, D. P., May, J. H., and Vargas, L. G. (2000). Modeling the uncertainty of surgical procedure times: Comparison of log- normal and normal models. *Anesthesiology*, 92(4):1160–1167.

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

[Vandenberghe et al., 2019] Vandenberghe, M., De Vuyst, S., Aghezzaf, E. H., and Bruneel, H. (2019). Surgery sequencing to minimize the expected maximum waiting time of emergent patients. *European Journal of Operational Research*, 275(3):971–982.

Appendices

A Extracted data

The appendix is purposely excluded for security reasons.