# Surgery planning for elective patients: A dedicated heuristic and an effective ALNS

Lahcene Mezouari, Jean-Paul Boufflet, Aziz Moukrim

# Surgery planning for elective patients: a dedicated heuristic and an effective ALNS

Lahcene Mezouari[a,*], Jean-Paul Boufflet[a], Aziz Moukrim[a]

[a] *Université de Technologie de Compiègne, CNRS, Heudiasyc (Heuristics and Diagnosis of Complex Systems) CS 60 319 - 60 203 Compiègne Cedex*

**Abstract**

Hospital organization, the medical concerns of the patient, surgery resources and the horizon to be considered are all elements that contribute to the variety of problems encountered in surgery planning. In this paper, we address the admission planning problem for which surgical interventions of hundreds of elective patients need to be scheduled months before the date of surgery. The health care surgery organization we consider here is based on a shared management of operating rooms and surgeons. The main issue for hospital planners is to schedule all the interventions under resource availability constraints while considering the patients' health priorities. We propose a two-phase 2PSC-EM randomized heuristic that obtains better results on literature benchmark instances. However, for some instances certain interventions are left unscheduled since straightforward heuristic failed to schedule all interventions. We investigated an effective Adaptive Large Neighborhood Search (ALNS) approach. Better results are obtained for each instance, all the patients' interventions are scheduled which had not been done before. The average improvement is about 11.2% and the processing times are shorter than the timeout fixed in the literature, except for one instance for which we succeeded to schedule all of the patients.

*Keywords:* Surgery planning, project scheduling, adaptive heuristic, operating room management, metaheuristic

## 1. Introduction

Health care systems are facing increasing demands in an environment of difficult budgetary constraints. Access to quality health services has become a major issue (Xie and Lawley, 2015). The surgery department represents an essential activity for the majority of hospitals, generating about two thirds of the hospital's income (Denton et al., 2007). About 52% of all hospital admissions are

*Corresponding author
*Email address:* lahcene.mezouari@hds.utc.fr (Lahcene Mezouari)

surgical interventions (Gupta, 2009). Given the budgetary constraints involved, increasing the number of surgeons and operating rooms, in addition to all of the other related resources, is not the solution to investigated first, but, instead the improved use of existing resources when feasible (Fischer et al., 2020). One way to do this is to build better surgery schedules with the aim of scheduling a maximum number of surgery interventions while taking limited resources with availability constraints into account.

Planning of surgical interventions has been widely studied, and the literature reviews (Rahimi and Gandomi, 2020; Samudra et al., 2016; Erdogan and Denton, 2011; May et al., 2011; Cardoen et al., 2010; Magerlein and Martin, 1978) provide an overview of the wide variety of problems encountered. For example, the hospital organization's stakeholders, the care processes, the hospital structure and the work regulations are all elements that contribute to the wide variety of problems. Surgeons may be fee-for-service or hospital employees, or both. In some cases, the hospital may allocate time windows for operating rooms to a department (e.g., a group of orthopedic surgeons) and leave it to the surgeons themselves to divide the time windows. In some other cases, surgeons schedule their own patients in operating rooms that they book a few weeks in advance. In yet other cases, provided that several surgeons have the surgery skills to perform an intervention that can be done in certain operating rooms, rooms and surgeons are viewed as shared renewable resources to be efficiently managed for the hospital organization. These cases can also be hybridized, and, consequently, public-private organizations are widely encountered.

Problems can be classified along several axes, such as patient characteristics or the length of the planning horizon (months, weeks, days). Patients are usually considered to be in one of two main categories: elective patients or non-elective patients. For the first, interventions can be planned months in advance by considering the main medical resource needs over the long term. For the second, interventions are either more urgent or unforeseen since a recent diagnosis for a patient may involve an intervention to be scheduled in the short term (several weeks) or immediately following an accident. In certain hospital organizations, emergency departments have their own operating rooms; if not, the sharing of operating rooms between the two categories of patients has to be considered. A typical way of managing these two types of patients at the same time is to reserve or to dedicate operating rooms either for elective or for non-elective patients, but it may depend on the horizon being considered. Patient admission planning may take only the main medical resource needs for an intervention into account, with the aim being to allocate a date for the surgical intervention for each patient, typically taking the long-term horizon (months) into consideration. Next, a plan for the following one or two weeks is drawn up on the basis of, first, the existing patient admission schedule and, second, the more urgent interventions of new patients that are only known a few weeks ahead of time. Finally, a schedule is drawn up every day for the next day. The weekly-based and daily-based schedules are usually subject to dynamic re-scheduling that should take cancellation or postponement policies for elective patients into account in order to disrupt patient admission planning as little as possible.

The health care surgery organization we consider here is based on a shared management of operating rooms and surgeons that are viewed as renewable resources available within certain time intervals, and, where each resource has a limited, constant, capacity. An intervention can be performed using one of several modes since several couples of operating rooms and surgeons can be used. A mode has a fixed duration that may depend on, for instance, the surgical act specific to the surgeon and to the operating room.

Riise et al. (2016) presented the GOSSP model (generalized operational surgery scheduling problem) as an extension of multi-project MRCPSP/Max, a generalization of the classical-resource constrained project scheduling problem (RCPSP), known to be NP-hard in the strong sense. The model proposed by the authors makes it possible to consider parameters that relate to patients (priority level, availability date, due date, hard due date) and resource availabilities, as well as to preferences that aim at scheduling certain interventions early in the morning. A surgeon may prefer to perform complex and long surgeries early in the morning. For some type of patients much more at risk or for children (it is hard for them to stay on an empty stomach) the surgery should be scheduled in the morning. Three planning situations have been modeled: the admission planning, the weekly problem and the daily problem. As the day of surgery approaches, more resources need to be managed and the model is able to integrate the information related to the planning situation. The authors proposed publicly available datasets. These datasets can be used to compare search methods. For the sake of generality, the authors have proposed an Adaptive Construct and Improve algorithm (ACI) to address all these planning situations without any off-line parameter tuning. To the best of our knowledge no other results have been published on these datasets. For more detailed insights we invite the reader to refer to Riise et al. (2016).

In this paper, we propose an effective approach dedicated to the admission planning problem for which the surgical interventions of hundreds of elective inpatients and outpatients need to be scheduled months before the date of surgery, considering that: (i) the main medical resource needs are the surgeons and the operating rooms; (ii) several surgeons have the surgery skills to perform the intervention that can be done in certain operating rooms; both rooms and surgeons are viewed as renewable resources; and several couples of operating rooms and surgeons can be used to schedule an intervention; (iii) each patient has a priority level, an availability date, a due date and a hard due date, and the surgery may have to be scheduled early in the morning.

The contributions of this work are summarized as follows:

- We propose a two-phase 2PSC-EM randomized heuristic based on the ideas of RCPSP constructive heuristics. We obtain better results for all the benchmark instances for which all the interventions can be scheduled by ACI. However, for some instances not all the patients' interventions are scheduled, the same for which the ACI algorithm also obtained unscheduled interventions. A more effective approach is needed.

- We investigate an ALNS based solution approach with the aim of schedul-

ing all the patients' interventions. The 2PSC-EM randomized heuristic is used to obtain initial solutions. The ALNS that we propose makes use of adaptive mechanisms for destruction and construction of solutions that have been investigated to deal with the resource limitations and the patients' medical constraints. An adaptive diversification mechanism associated with certain destruction operators permits us to broaden the search around a solution in order to obtain better solutions. We carefully designed the acceptance of a lower-quality solution to obtain a good trade-off with the adaptive diversification mechanism.

- Preliminary experiments were done to tune the parameters of the components of the ALNS to obtain a good trade-off between solution quality and processing time. We conducted computational experiments to investigate the contribution of each component of the ALNS approach to show when each of them are beneficial.

- Better results are obtained for each instance, all the patients' interventions are scheduled which had not been done by the ACI algorithm. The average improvement is about 11.2% and the processing times are shorter than the timeout fixed for the ACI algorithm experiments except for one instance for which the ALNS succeeded to schedule all of the patients. The ALNS is an approach that automatically adapts to the difficulty of the instance. The ALNS takes little time for the majority of the instances, and it schedules all the patients' interventions within a reasonable processing time for more difficult instances. The ALNS approach outperforms the results of a more general approach like ACI.

The remainder of this paper is organized as follows. Section 2 is a review of the literature from the short-term horizon of a day, up to the middle-term horizon of several weeks, which we put into the perspective of the admission planning problem over several months. Section 3 presents the admission problem we address here. Constraints, parameters and objective function related to patients are detailed. Section 4 presents the two-stage 2PSC-EM heuristic that we propose. Our ALNS approach is described in Section 5 and several components are explained in detail. The computational experiments are reported and commented on in Section 6. The conclusion and perspectives for further development are to be found in Section 7.

## 2. Literature review

The planning and scheduling of patients for surgical interventions have been widely investigated in the literature. In this study we consider the deterministic case, for non deterministic approaches we invite the reader to refer to recent works as presented in Zhou et al. (2021).

Metaheuristic based solution approaches have been extensively reported in literature to address a large variety of optimization problems, for a comprehensive survey we invite the reader to refer to Hussain et al. (2019). Recently, Lan

et al. (2021) presented a survey on the applications of Variable neighborhood search (VNS) in the health care area. In addition to VNS, the ALNS approach makes use of adaptive mechanisms.

For a more general and recent review on the ALNS metaheuristic framework and its applications, we invite the reader to refer to Mara et al. (2022).

We review here studies regarding the scheduling of surgical interventions from a scheduling horizon perspective, ranging from the short-term horizon of a day, up to the middle-term horizon of several week. Short-term and middle-term problems have many specific features that depend on the practical cases addressed by the study, as well as aspects that can be viewed in a longer-term perspective.

**Daily intervention scheduling problems**

Park et al. (2021) proposed a mixed integer program and a heuristic based on the grouping of interventions of the same surgeon for the surgical planning of the daily interventions of a Korean university hospital. Each patient is already assigned to a surgeon. Some operating rooms are reserved for urgent surgeries, beyond the scope of the daily schedule. The surgeons' preferences for operating rooms as well as cooperative surgery constraints, where several surgeons simultaneously or sequentially perform an operation are considered. The aim is to minimize the total number of overtime hours, the number of operating rooms used, the number of surgeries not allocated to preferred rooms and the number of surgeries allocated to unfavorable operating rooms.

Wang et al. (2015) proposed two approaches to address the daily surgical intervention scheduling problem of a Belgian university hospital. The interventions to be scheduled per day are known. The authors considered the constraints of operating room and surgeon availability, as well as the limited number of recovery beds. Some interventions should start earlier than others, depending on the priorities related to patient status (e.g., children, diabetics). The first approach uses a mixed integer programming model, whereas the second uses constraint programming. The main aim is to minimize the makespan of the scheduled interventions.

Xiang et al. (2015) proposed a solution approach based on Ant Colony Optimization (ACO) to address the daily surgical intervention scheduling problem of the MD Anderson Medical Center in Houston, Texas (USA). The surgeries to be scheduled on a daily basis are known. The aim is to minimize makespan considering the limited amount of resources (e.g., surgeon, anesthesiologist, nurse, beds) that are needed to perform a surgery, as well as the pre-surgery and post-anesthesia stages. Xiang (2017) proposed an extension of the ACO approach considering two additional aims to minimize the variation in the working time of resources and to minimize the total overtime of all resources.

Sier et al. (1997) proposed a solution for a daily surgical intervention problem based on a simulated annealing algorithm. A mixed integer nonlinear formulation for the studied problem was proposed. The objective function is a weighted sum of all problem constraints. The feasibility of the solution is not guaranteed, the approach aims at minimizing the constraints violations.

The daily surgical intervention scheduling problems have common features that can be observed, regardless of the specific organization. The list of elective patients' interventions to be scheduled is known and all the interventions are scheduled. The makespan or the overtime hours are often considered as an explicit part of the objective function since it is important not to exceed normal working hours. Since all interventions must be scheduled, the uses of the critical resources to make a daily schedule feasible are to be considered since each resource has a limited, and constant, capacity.

**Weekly intervention scheduling problems**

Roshanaei and Naderi (2021) proposed a sequence-based mixed-integer programming model and a constraint programming model to address a problem initially proposed in Hashemi Doulabi et al. (2016) to plan surgical interventions over a week horizon. Patients are managed using two sets (mandatory and optional patients). The patient-to-day allocation, the (patient, day)-to-operating room assignment, the (patient, day, operating room)-to-surgeon, and, finally, the scheduling of the start times of these surgeries are considered in turn. The aim is to maximize the total scheduled surgical time. Given that this problem can be hierarchically structured, the authors investigated Benders decomposition that makes it possible to achieve better results.

Akbarzadeh et al. (2020) considered a surgery planning problem over several days with nurse re-rostering. Experiments were conducted on generated data based on real data from the Sina Hospital (Tehran, Iran). The heuristic solution approach proposed first builds an optimal LP solution obtained via a column generation algorithm, after which a diving heuristic is applied to drive fractional solutions to integrality. The aim is to maximize the use of the operating rooms by considering the nurses' needs as well.

Oliveira et al. (2020) investigated patient scheduling over two weeks considering their prioritization and available sessions (operating rooms and surgeons) for a Urology Department at a university hospital in Quebec City. A MIP model is used on randomly generated instances based on real data to study the impact of several policies for managing wait-listed patients (four categories of patients are considered by urgency level).

Ballestín et al. (2019) considered the scheduling and rescheduling of elective patients over two weeks. A tentative schedule is first built, followed by a final schedule that is calculated some days before the start of the planning horizon in order to take new information (change in the waiting list or in the status of the patients) into account. The aims are to minimize the number of tardy patients and to maximize the utilization rate of the operating rooms. A linear model formulation is used to perform simulations on randomly generated instances based on real data from a Spanish hospital. The authors studied the trade-off between the degree of the changes allowed in the tentative schedule and the benefits in relation to the objective function.

Zhang et al. (2019) considered a weekly scheduling single-specialty elective surgery problem. The operating rooms and surgical intensive care unit have a limited capacity. However, the surgeons are not considered in this study. A

two-level approach was proposed. By applying a Markov decision process, the first level selects interventions from a waiting list of patients. The second level is based on an approximate dynamic programming approach to assign selected interventions to rooms.

Castro and Marques (2015) addressed the problem of planning over a week of elective surgeries of different specialties to be selected from among a large list of elective surgeries on the basis of three priority levels. There is a maximum weekly operating time limit for every surgeon. A two-step decomposition approach that makes use of Generalized Disjunctive Programming (GDP) is proposed. The first step generates a planning model that does not consider the surgeon availability constraints. The second step generates a scheduling model, that aims at determining the start time of the selected interventions by assigning the surgeons to the operating rooms, and taking the types of interventions and the availability of the surgeons into account. Some surgery/surgeon/room assignments cannot be translated into a feasible schedule due to conflicts not considered in the planning model, and some interventions of the subset may consequently be left unscheduled. Room and day assignments do not change but some lower-priority planned surgeries can be moved back to a waiting list. The proposed approach is tested on real data from central and university hospital in Lisbon (Portugal). The aim is to maximize the total surgical time.

The weekly surgical interventions scheduling problems have common features that can be observed, regardless of the specific organization. The main resources to be managed are the surgeons and the operating rooms. Considering a hospital organization and given the relative importance of the stakeholders, the constraints on the allocations of the surgeons and of the operating rooms need to be adequately managed. Not all interventions can be planned because the waiting list exceeds the limit of the considered horizon of a few days, a week or a few weeks.

Mathematical programming approaches are popularly used to address surgical intervention scheduling problems over a day or a week. One reason for their success is that mathematical programming offers a framework that is powerful to express the wide variety of the encountered context dependent constraints. However, as the problems grow in size or in complexity/number of the different constraints, solvers may face difficulties in attaining solutions within reasonable processing times. Heuristic approaches based on mathematical formulations, on dedicated heuristics or on meta-heuristics are used to obtain good solutions.

Regardless of the specific features of the organization of surgery rooms, the common features are the list of patients, each having a priority, and, the main resources that are the surgeons and the operating rooms with their availabilities.

The more the horizon increases, the more the importance of scheduling all patients increases while considering their priority in terms of access to surgery. Given a large list of patients for whom interventions need to be scheduled, each with his or her own health characteristics, it is an issue for a hospital to know whether it is feasible to have an admission plan where all interventions can be scheduled. Not only can it be useful to determine a day for surgery, but an

hour as well, allowing constraints such as scheduling certain interventions in the morning to be taken into account as early as possible. The surgeons and the operating rooms can be considered on a long-term horizon provided that the availability constraints are known.

| No. | Literature | Problem characteristics | | | | Optimization method |
|---|---|---|---|---|---|---|
| | | ORs | Surg | Obj | Data | |
| 1 | Park et al. (2021) | ✓ | ✓ | SO | BRD | EA: MIP, H |
| 2 | Roshanaei and Naderi (2021) | ✓ | ✓ | SO | TD | EA: MIP, EA: CP |
| 3 | Akbarzadeh et al. (2020) | ✓ | | SO | BRD | H |
| 4 | Oliveira et al. (2020) | ✓ | ✓ | SO | BRD | EA: MIP |
| 5 | Ballestín et al. (2019) | ✓ | ✓ | SO | BRD | EA: MIP, H: S |
| 6 | Zhang et al. (2019) | ✓ | | SO | TD | H: ADP |
| 7 | Xiang (2017) | ✓ | ✓ | MO | TD | MH: ACO |
| 8 | Hashemi Doulabi et al. (2016) | ✓ | ✓ | SO | TD | EA: CG |
| 9 | Riise et al. (2016) | ✓ | ✓ | SO | BRD | H |
| 10 | Wang et al. (2015) | ✓ | ✓ | SO | BRD | EA: MIP, EA: CP |
| 11 | Xiang et al. (2015) | ✓ | ✓ | SO | TD | MH: ACO |
| 12 | Castro and Marques (2015) | ✓ | ✓ | SO | RD | H: GDP |
| 13 | Sier et al. (1997) | ✓ | | SO | BRD | H |
| 14 | this study | ✓ | ✓ | SO | BRD | MH: ALNS, H |

Note :

- ORs(Operating Rooms), Surg(Surgeons), Obj(Objective).
- Obj: SO(Single-Objective), MO(Multi-Objective).
- Data: RD(Real Data), BRD(Based on Real Data), TD(Theoretic Data).
- Optimization method: H(Heuristics), MH(Metaheuristics), EA(Exact Algortithm).
- CP(Constraint Programming), MIP(Mixed Integer Program).
- ACO(Ant Colony Optimization), GDP(Generalized Disjunctive Programming).
- S(Simulation), ADP(Approximate Dynamic Programming).

Table 1: Related studies on surgical intervention scheduling problems.

In Table 1 we provide a synthetic overview of the literature that relates to our study by chronological order.

For a more detailed overview of the literature about surgery planning problems, we invite the reader to refer to Rahimi and Gandomi (2020), Samudra et al. (2016), Erdogan and Denton (2011), May et al. (2011), Cardoen et al. (2010), Magerlein and Martin (1978) who provide reviews on variants of these problems and on solution approaches.

## 3. Problem description and MIP model

Hospitals need to create the admission plans for surgical interventions of patients months before the date of surgery in consideration of their most critical resources: the surgeons and the operating rooms. In this paper, we consider elective inpatients and outpatients for whom a surgical intervention is to be

scheduled regardless of the events that may arise in shorter time horizon schedules.

We focus on the following admission planning problem: given a set of hundreds of patients for whom surgical interventions need to be scheduled over several months, the main issue is to schedule all the interventions in order to obtain a start date (a day and a time) for each intervention. There may be interventions that cannot be feasibly scheduled, but this should be avoided as much as possible.

We first present the data and parameters that relate to patients (priority level, availability date, due date, hard due date), the resource availabilities, as well as preferences that aim at scheduling certain interventions early in the morning. Next, we report the mathematical formulation of the problem from Riise and Mannino (2012) and Riise et al. (2016). In the following, we use most of the notations introduced in Riise and Mannino (2012) and Riise et al. (2016).

**Data and parameters**

Patients, and some related medical and practical issues, are considered using the following data and parameters:

$H$, planning horizon, or planning period length;

$z$, the number of time units in a day;

$P$, set of elective patients for whom a surgical intervention has to be planned, indices $p$, we denote $|P|$ as its size;

$X_p$, earliest start date for the intervention of patient $p$;

$D_p$, due date; patient $p$ should preferably be scheduled before;

$H_p$, hard due date; patient $p$ must be scheduled before; otherwise, the intervention is unscheduled;

$\beta_p$, for a patient, the relative importance to schedule the patient's surgical intervention;

$P^{em} \subset P$, subset of patients for whom the intervention should be planned **e**arly in the **m**orning, we denote $|P^{em}|$ as its size;

$\tau$, for all patients $p \in P^{em}$, the time in the morning from which the intervention can be scheduled;

$\gamma_p$, for a patient $p \in P^{em}$, the relative importance to schedule the patient early in the morning;

$R_s$, set of surgeons, indices $s$;

$R_r$, set of operating rooms, indices $r$;

$R$, set of all resources, $R = R_s \cup R_r$;

$R^p$, set of resources assigned to the surgical interventions of patient $p$, through the choice of mode;

$K_i$, set of successive disjoint availability intervals associated with resource $i$ (surgeon or room);

$K_i^p \subseteq K_i$, set of intervals for which resource $i$ is available to the surgical interventions of patient $p$;

$M$, set of modes $m = (r_s, r_r) \in R_s \times R_r$, $r_s$ indice of a surgeon and $r_r$ indice of a room; these are all the combinations of surgeons and operating rooms;

$M_p$, set of feasible modes $m$ for the surgical intervention of patient $p$;

$\varphi^k$, fixed starting time of resource interval $k$;

$\sigma^k$, latest end time of resource interval $k$;

$u_i^m$, if mode $m$ uses resource $i$ then $u_i^m = 1$;

$\vartheta_p^m$, length of the surgical intervention for patient $p$ when using mode $m$.

Each patient $p$ has an earliest start date $X_p$ for the intervention and a due date $D_p$ defined to reflect a medical priority. Each patient $p$ has a hard due date $H_p \geq D_p$. The starting date of the intervention should be strictly below this hard due date; otherwise, the surgical intervention is left unscheduled. Parameters $X_p$, $D_p$, and $H_p$ are integer numbers of days relative to the beginning of the planning horizon. For a patient $p$, the value $\beta_p$ is linked to the patient's degree of priority for the surgical intervention.

For some particular patients $p \in P^{em}$, the surgical intervention should be preferably scheduled **e**arly in the **m**orning; the parameter $\tau$ is the time in the morning from which the surgical intervention can be scheduled. The parameter $\gamma_p$ is the relative importance to schedule the patient's surgical intervention earlier in the morning.

In order to be scheduled, a surgical intervention requires a surgeon and a room. The sets $R_s$, $R_r$, $R$ and $R^p$, are used to manage the resources. The sets $K_i$ are useful to take the daily working hours of the teams or the surgeons into consideration, as well as the operating room hours and the maintenance operations of these surgical technical platforms.

For a patient $p$, the set of feasible modes is $M_p$, depending on the surgical operation to be performed. The medical evaluations during the preoperative consultations make it possible to select surgeons not only in accordance with the needed specialty, but also with the professional experience that is related to the difficulty of the medical case. Usually in such organization, the surgeons themselves, organized by specialty, decide who can perform the surgical operation. Consequently, the length $\vartheta_p^m$ of the surgery depends on the surgeon and on the operating room where it can be done.

One mode $m$ should be chosen to plan the surgical intervention of patient $p$ from among $M_p$. The preemption of surgical interventions is not authorized.

Given that we are scheduling many patients, when using mode $m = (r_s, r_r)$ for a patient $p$, an interval $\mathcal{I}$ from among the set of intervals $K_{r_s} \cap K_{r_r}$ is to be chosen to make it possible to plan the surgical intervention of length $\vartheta_p^m$, taking all other surgical interventions that are already scheduled into account.

For a patient $p$, a surgical intervention is to be scheduled; an intervention corresponds to one patient $p$. For the sake of simplicity, we use the same indices $p$ for indexing interventions in the sequel.

**Mathematical formulation**

We introduce the mathematical formulation of the problem from Riise and Mannino (2012) and Riise et al. (2016). The variables are as follows:

$x_p^m = 1$, if the surgical intervention of patient $p$ uses mode $m \in M_p$, and 0 otherwise;

$q_p^k = 1$, if the surgical intervention of patient $p$ uses resource interval $k \in K_i$, for resource $i$, and 0 otherwise;

$t_p$, the non-negative starting time of the surgical interventions of patient $p$;

$z_{pp'} = 1$, if the surgical intervention of patient $p$ precedes the surgical interventions of patient $p'$, and 0 otherwise;

$c_p$, completion time of the surgical interventions of patient $p$;

$g_p^i = 1$, if the surgical intervention of patient $p$ uses resource $i$ in the chosen mode, and 0 otherwise;

$h^p = 1$, if the surgical intervention of patient $p$ is unscheduled, and 0 otherwise.

The MIP formulation is as follows:

Min

$$W_{un} \underbrace{\frac{1}{\sum_{p \in P} \beta_p} \sum_{p \in P} \beta_p h^p}_{O_{un}} \tag{1a}$$

$$+ W_{wt} \underbrace{\frac{1}{1 + \lambda |P|} \sum_{p \in P} o_{wt(p)} (1 - h^p)}_{O_{wt}} \tag{1b}$$

$$+ W_{em} \underbrace{\frac{1}{z |P^{em}|} \sum_{p \in P^{em}} \gamma_p \Gamma(p) (1 - h^p)}_{O_{em}} \tag{1c}$$

Subject to:

$$x_p^m \in \{0, 1\} \quad \forall p \in P, \ \forall m \in M_p \tag{2}$$

$$z_{pp'} \in \{0,1\} \quad \forall p,p' \in P \tag{3}$$

$$t_p \in \mathbb{R}_+ \quad \forall p \in P \tag{4}$$

$$q_p^k \in \{0,1\} \quad \forall i \in R,\ \forall k \in K_i^p \tag{5}$$

$$c_p \in \mathbb{R}_+ \quad \forall p \in P \tag{6}$$

$$g_p^i \in \{0,1\} \quad \forall i \in R,\ \forall p \in P \tag{7}$$

$$c_p = t_p + \sum_{m \in M_p} \vartheta_p^m x_p^m \quad \forall p \in P \tag{8}$$

$$g_p^i = \sum_{m \in M_p} u_i^m x_p^m \quad \forall i \in R,\ \forall p \in P \tag{9}$$

$$\sum_{m \in M_p} x_p^m = 1 - h^p \quad \forall p \in P \tag{10}$$

$$\sum_{k \in \mathcal{K}_i^p} q_p^k = g_p^i \quad \forall p \in P,\ \forall i \in R \tag{11}$$

$$z_{pp'} + z_{p'p} \leq 1 \quad \forall p,p' \in P \tag{12}$$

$$t_{p'} - c_p \geq (z_{pp'} - 1)\mathcal{M} \quad \forall p,p' \in P \tag{13}$$

$$z_{pp'} + z_{p'p} \geq g_p^i + g_{p'}^i - 1 \quad \forall p,p' \in P,\ \forall i \in R \tag{14}$$

$$t_p \geq X_p \quad \forall p \in P \tag{15}$$

$$c_p \leq H_p \quad \forall p \in P \tag{16}$$

$$t_p - \varphi^k q_p^k \geq 0 \quad \forall p \in P,\ \forall i \in R,\ \forall k \in K_i^p \tag{17}$$

$$c_p - \sigma^k q_p^k - (1 - q_p^k)\mathcal{M} \leq 0 \quad \forall p \in P,\ \forall i \in R,\ \forall k \in K_i^p \tag{18}$$

Constraints (2)-(7) define the variables. Constraint (8) sets the completion times $c_p$. Constraint (9) ensures resource $i$ is used in the chosen mode $m$ for patient $p$. Constraint (10) ensures that at most one mode $m$ is chosen and set $h^p$ accordingly. Constraint (11) ensures that one interval $k$ is used for each resource $i$ of mode $m$ for a patient $p$. Constraints (12), (13) and (14) ensure precedence

constraints when two patients $p, p'$ use a same resource $i$. Constraints (15) and (16) ensure surgical intervention to be in the required interval $[X_p, H_p]$ for patient $p$. Constraints (17) and (18) ensure surgical intervention interval $[t_p, c_p]$ to be inside interval $[\varphi^k, \sigma^k]$ of resource $i$. In constraints (13) and (18), the big-M value is $\mathcal{M} = H$ (see Riise and Mannino (2012)).

A feasible solution $S$ is composed of $\widetilde{P} \subseteq P$, a set of elective patients for whom a surgical intervention is scheduled ($h^p = 0$) and a set $P \setminus \widetilde{P}$ of patients for whom a surgical intervention is not scheduled ($h^p = 1$). In the same way, we consider $\widetilde{P}^{em}$ and $P^{em} \setminus \widetilde{P}^{em}$.

The objective function to be minimized is a weighted sum of three normalized terms, see (1a)-(1c). Given a feasible solution, the first term $O_{un}$ of the objective function aims at minimizing the number of elective patients for whom the surgical interventions are left **un**scheduled. The term is a normalized sum of the $\beta_p$ values, the relative importance to schedule the patient's surgical intervention, for all the elective patients for whom their surgical intervention ($p \in P \setminus \widetilde{P}$) cannot be scheduled.

The second term of the objective function $O_{wt}$ is a normalized sum of the **w**aiting **t**imes for the patients for whom the surgical intervention is scheduled. For a patient $p$, the **w**aiting **t**ime $o_{wt(p)}$ is assessed using the following function:

$$o_{wt(p)} = \begin{cases} \frac{c_p - X_p}{D_p - X_p} & \text{if } c_p \leq D_p \\ 1 + \lambda \frac{c_p - D_p}{H_p - D_p} & \text{otherwise.} \end{cases} \tag{19}$$

Overall, for a patient $p$, this piecewise linear function increases as $c_p$ increases relative to the earliest start date $X_p$ and relative to the due date $D_p$. Given that we should schedule the surgery before the due date $D_p$, the slope value becomes a steeper value $\lambda$ when $c_p > D_p$. The purpose is to over-penalize the surgical intervention of a patient $p$ that is scheduled after the due date $D_p$.

The third term $O_{em}$ is a normalized sum that evaluates the surgical interventions of patients $p \in P^{em}$ whose interventions should be planned **e**arly in the **m**orning. Given that $\tau$ is known for all patients $p \in P^{em}$, the function $\Gamma(p)$ computes the difference of time between $\tau$ and $c_p$ for the chosen planning day; this value is weighted by the relative importance $\gamma_p$.

The normalized terms are weighted by $W_{un}$, $W_{wt}$ and $W_{em}$. We have $W_{un} > W_{wt} > W_{em}$, which clearly reflects the hierarchy for medical considerations: the priority is to, first, plan as many patients' interventions as possible; second, as close as possible to the due date; and, finally, to schedule in the morning the patients for whom this constraint should be respected.

We denote $Obj(S)$ as the evaluation of a solution $S$ that we compute as presented in equations (1a)-(1c).

## 4. The 2PSC-EM heuristic

Given that surgical interventions of the set $P$ of patients can be viewed as projects that require renewable and shared resources, the admission planning

problem is presented in Riise et al. (2016) as a multi-project/multi-mode RCPSP with minimum and maximum time lags.

We propose a Two Priorities Scheduling Construction phase (2PSC) based on the ideas of RCPSP constructive heuristics (see Kolisch and Hartmann (1999); Tormos and Lova (2003); Almeida et al. (2016)), followed by an improvement phase that focuses on patients that need to be scheduled **e**arly in the **m**orning. We denote this heuristic as 2PSC-EM.

The 2PSC construction phase uses two priority lists, one for managing the patients to be scheduled ($P$), and the second for managing the execution modes of each patient ($M_p$). The general idea is to increase the time at each iteration in order to choose the next surgical intervention to be planned by taking all the surgical interventions in progress that are already planned into account (these are assumed to still be in progress) since they cannot be interrupted (no preemption allowed).

**Algorithm 1:** The construction phase 2PSC

**Input** : $P$ set of interventions (and patients), $R_s, R_r$ sets of resources (surgeon, room)
**Output** : $S$ solution, computed planning, $\mathcal{L}_u$ list of interventions left unscheduled
**Variables** : $t_i$ time where a surgical intervention can be scheduled; $i$ number of iterations, relative to time $t_i$; $C_i$ set of interventions, completed at time $t_i$; $A_i$ set of interventions, in progress at time $t_i$; $R_i$ set of resources used by interventions in $A_i$; $\mathcal{L}_i$ list of interventions that can be scheduled at time $t_i$; $p$ number of interventions; $M_{p,i}$ list of modes that can be used at time $t_i$ for intervention $p$; $m$ a mode $(r_s, r_r)$; $E_p$ end time of the intervention $p$

```
1  i, t_i ← 0
2  S, A_0, C_0 ← ∅
3  R_0 ← R_s ∪ R_r
4  while |A_i ∪ C_i| < |P| ∧ (interventions can be scheduled without exceeding the horizon)
     do
       // time of the intervention that is in progress with the soonest end
5      t_i ← min_{p∈A_i}{E_p}
6      Compute using timestep i number: C_i, A_i, R_i, L_i
7      Sort L_i in increasing order of D_p  // earliest due date
8      Shuffle L_i by block of equal D_p values
9      while L_i ≠ nil do
10       p ← L_i.pop_front()  // first intervention that can be scheduled at time t_i
           // One intervention having one of the smallest D_p value is selected
11       Compute M_{p,i}  // usable modes m a time t_i for intervention p
12       Sort M_{p,i} in increasing order of ϑ_p^m  // small length first
13       m ← M_{p,i}.pop_front()  // mode with smallest length ϑ_p^m
14       Schedule intervention p at t_i with mode m in solution S
15       E_p ← t_i + ϑ_p^m
16       Update R_i, A_i, L_i  // other intervention may be scheduled at t_i
17       Shuffle L_i by block of equal D_p values
18 Compute L_u  // List of interventions left unscheduled
19 return S, L_u
```

The algorithm of the 2PSC construction phase is shown in Algorithm 1. The iteration $i$ corresponds to the time $t_i$ where a surgical intervention can be scheduled; they are initialized at $i = 0$ and $t_0 = 0$. The beginning of the planning horizon is 0 and the end is $t_{end}$.

We denote $E_p$ as the **E**nd time of intervention $p$ when surgical intervention is scheduled for a patient $p$ with length $\vartheta_p^m$ using a mode $m$. We denote $C_i$ as the

set of the surgical interventions of patients that have been **C**ompleted at time $t_i$, formally $C_i = \{p \in P, E_p \leq t_i\}$. We denote $A_i$ as the set of **A**ctive surgical interventions of patients at time $t_i$, formally $A_i = \{p \in P, E_p - \vartheta_p^m \leq t_i < E_p\}$; these surgical interventions are in progress. The surgical interventions that have been planned up to $t_i$ belong either to $C_i$ or to $A_i$. We denote $R_i$ as the set of resources of $R_s$ or $R_r$ used by the active interventions in the set $A_i$. We denote $\mathcal{L}_i$ as the list of interventions that can be scheduled at time $t_i$. The interventions in $\mathcal{L}_i$ are $\{p \in P \wedge p \notin A_i \wedge p \notin C_i, X_p \leq t_i\}$. These lists manage the interventions of patients to be scheduled throughout the algorithm.

An intervention for a patient $p$ from among $\mathcal{L}_i$ can possibly be performed using mode $m$ from among $M_p$. However, not all these modes are possible at time $t_i$, since resources may already be used by some interventions in $A_i$ that are not finished. We denote $M_{p,i}$ as the list of modes that can be used at time $t_i$. The modes in $M_{p,i}$ are: $\{m = (r_s, r_r) \in M_p, (r_s, r_r \notin R_i) \wedge (t_i + \vartheta_p^m \leq H_p) \wedge ([t_i, t_i + \vartheta_p^m] \in K_{r_s} \cap K_{r_r})\}$.

In the first loop of Algorithm 1, the overall idea is to manage the interventions to be planned by applying the earliest due date rule. We first update $t_i$ by selecting the time of the intervention that is in progress with the soonest end in $A_i$, computed at the previous iteration. Next, the new $C_i, A_i, R_i$ and $\mathcal{L}_i$ that correspond to the new timestep $i$ are computed. The list $\mathcal{L}_i$, the interventions that can be scheduled at time $t_i$, is then sorted in increasing order of $D_p$; the priority given to the earliest due date aims at minimizing the $O_{wt}$ term. The loop ends when all the interventions are scheduled or when no more interventions can be scheduled without exceeding $t_{end}$.

In the second loop of Algorithm 1, the idea is to schedule as many interventions as possible at time $t_i$ by applying the shortest processing time rule. The first intervention $p$ that can be scheduled at time $t_i$ is selected. Its usable modes $M_{p,i}$ at time $t_i$ are computed and then sorted by increasing length $\vartheta_p^m$. Next, the mode $m$ with the smallest length is chosen, the intervention $p$ is scheduled, and its end time $E_p$ is updated. The sets and the list $S$, $R_i$, $A_i$ and $\mathcal{L}_i$ are updated. For a scheduled intervention $p$, we set $c_p = t_i$ for the intervention start time. The loop ends when no more interventions can be scheduled at time $t_i$. The rule aims at freeing the resources as soon as possible.

We observed that patients may have the same values of due date $D_p$. We can take advantage of this observation by introducing a partial randomization to obtain different solutions when performing several runs. We shuffle the consecutive blocks of $\mathcal{L}_i$ that correspond to the same value of $D_p$ (see lines 9 and 19), and by doing so, the interventions with equal $D_p$ are chosen at random.

The 2PSC algorithm computes a solution $S$, so $\widetilde{P}$ is known. The interventions that have been left **u**nscheduled are computed and we return $\mathcal{L}_u$. This heuristic approach aims at minimizing the $O_{wt}$ term, but can be improved by considering the $O_{em}$ term.

We propose an improvement phase EM with the aim of improving the number of interventions of patients in $\widetilde{P}^{em}$ who are scheduled early in the morning. We try to move interventions $p \in \widetilde{P}^{em}$ a step backwards to schedule them early

in the morning. For each patient $p \in \widetilde{P}^{em}$, we obtain the day $d$ where the surgical intervention is scheduled and then we obtain the list of all the scheduled interventions of this day. This list $\mathcal{L}_d$ is sorted in increasing order of the time element of $c_p$. Given that $p$ is in position $i$, we try to swap intervention $p$ with each intervention scheduled before. Function $Swap(s, i, j)$ returns True when it is possible to exchange the interventions. Provided that the two modes and the two lengths are the same, the exchange is performed by this function and $S$ is updated. This straightforward condition has a major advantage, i.e., we can obtain an improved solution at small expense by avoiding the computing time necessary to re-schedule part of the solution.

Given a solution $S$ obtained by applying EM, the set $\widetilde{P}$ is the same, and for every day $d$, the set of scheduled interventions is still the same. We recall that parameters $X_p$, $D_p$, $H_p$ and the number of days in $c_p$ are integer numbers of days relative to the beginning of the planning horizon. Consequently, the first two terms of the objective function do not change. The swap of the modes of execution of the interventions leads to an equivalent solution relative to the constraints of the problem (resource consumption and release). The start time of some interventions of $\widetilde{P}^{em}$ may change, which may decrease the third term of the objective function.

The 2PSC-EM that we propose is a fast dedicated heuristic that computes a schedule that complies with the resource usage constraints, while addressing medical considerations that are related to patients. By introducing the shuffling of list $\mathcal{L}_i$, it can be run many time to obtain different solutions.

## 5. ALNS metaheuristic for the admission planning problem

The 2PSC-EM heuristic may compute a solution of good quality but, some interventions may be left unscheduled. It is of importance for the hospital management to know whether the interventions for all the elective patients can be scheduled.

The ALNS approach was presented for the first time by Ropke and Pisinger (2006). We provide a contextualized overview that consider the planning admission problem addressed here. We then present the general structure of our Adaptive Large Neighborhood Search (ALNS) in Section 5.1 .

For a large-scale neighborhood search, similar solutions are obtained through modifications to the original solution by applying several local search operators. Given that some interventions may be left unscheduled, we explore the neighborhood of a current solution by first applying a destruction operator to free resources. Next, all the interventions that remain to be planned are scheduled by applying a repair operator with the aim of scheduling all the interventions whenever possible. We present the four destruction operators and the three repair operators that we investigate in Sections 5.2 and 5.3. The degree of destruction needs to be tuned since it affects diversification. An adaptive diversification mechanism is used to manage the degree of destruction for some destroy operators. A couple of destruction and repair operators plays the role of a neighborhood for the local search.

The success of operators may vary depending on the problem instance. An adaptive choice generally leads to better results rather than fixing the choice of operators for the entire course of the algorithm. The adaptive mechanism presented in Section 5.4 is based on a roulette wheel principle that we use to manage the success of destruction operators and the success of repair operators. Each operator has a weight that represents its share in a wheel. The weights are updated according to the performance; efficient operators are used more often than less efficient operators.

The ALNS makes use of these destroy/repair operators to explore the neighborhood of a solution. However, that may not be sufficient to allow the search process to avoid or escape local optima. The acceptance method of ALNS has the purpose of deciding to either continue exploring the neighborhood of a current solution or to select a newly created one. A lower-quality solution can be accepted with the aim of escaping from local optima. We present an adaptive acceptance strategy in Section 5.5 based on the record-to-record travel algorithm, originally proposed by Dueck (1993).

The ALNS that we propose makes use of adaptive mechanisms for destruction, construction and acceptance procedures. Several parameters need to be tuned to obtain a good efficiency of these adaptive mechanisms.

### *5.1. General structure of ALNS*

Algorithm 2 gives the general structure of the ALNS that we outline here before providing more detailed insights.

Given an initial solution $S$ and the list $\mathcal{L}_u$ of interventions left unscheduled, both computed by the 2PSC-EM heuristic (see Section 4), the algorithm alternates destruction and repair phases on a current solution $S_{cur}$ to compute an incumbent solution $S_i$ (lines 5-10). The incumbent solution is denoted as $S_i$. However, we do not create a new container for every incumbent solution for each iteration $i$, where index $i$ is used for explanation purposes. When the incumbent solution $S_i$ fulfills the acceptance criteria, it is retained as the new current solution $S_{cur}$, and, the number of iterations $i$ is reset to zero; otherwise it is incremented (lines 11-15). The best solution found so far within iterations $S_{best}$ is kept (lines 16-18). Provided that a maximum number of iterations $Iter_{max}$ without any improvement is met, the ALNS algorithm stops and then returns $S_{best}$ and $\mathcal{L}_u$.

The destruction operator $O_{des}$ is selected from among a set of destruction operators (line 5) that are either day-oriented or intervention/patient-oriented (see Section 5.2). For intervention/patient oriented destruction operators ($DayDestroy(O_{des})$=false), a random number of $k$ interventions, bounded by $D_{max}$, are to be removed (line 7). An adaptive diversification mechanism manages $D_{max}$ (line 20), which makes it possible to broaden the search around the best solution found so far, with the intention of finding a better solution (see Section 5.2). The destruction operator is performed using $Destruction(S_{cur}, k, O_{des})$ (line 8). This frees resources but an additional list $\mathcal{L}_{des}$ of interventions are unscheduled.

**Algorithm 2:** Adaptive Large Neighborhood Search (ALNS) algorithm

```
Input      : S_0 initial solution, L_u list of interventions left unscheduled
Output     : S_best the best solution found, L_u list of interventions left unscheduled
Parameters: D_limit limit for diversification degree
            Iter_max maximum number of iterations without any improvement
Variables  : S_cur current solution, i number of iterations
            D_max maximum number of interventions to remove on several days
            O_des selected destruction operator, O_rep selected repair operator
            k number of interventions to remove on several days
            S_i incumbent solution
1  S_best, S_cur ← S_0  // Initialize best and current solution
2  i ← 0
3  D_max ← 3
4  while i < Iter_max do
5      O_des ← SelectDestructionOperator()  // select a destruction operator
6      k ← 1
7      if ¬DayDestroy(O_des) then  k ← rand(1, D_max)
        // remove k interventions on several days, or all interventions of a day
8      (S_i, L_des) ← Destruction(S_cur, k, O_des)
9      O_rep ← SelectRepairOperator()  // select a repair operator
10     S_i ← Construction(S_i, L_des, L_u, O_rep) // insert as many interventions as
        possible
       // Acceptance procedure on S_i, incumbent solution
11     if Accept(S_best, S_i) then
12         S_cur ← S_i // S_i is accepted as the new current solution
13         i ← 0
14     else
15         i ← i + 1
16     if Obj(S_i) < Obj(S_best) then
17         S_best ← S_i // A new best solution is found
18         D_max ← 3
19     else
           // Update maximum adaptive diversification degree
20         if ¬DayDestroy(O_des) then  D_max ← min(D_max + 1, D_limit)
       // Update weights of the two roulette wheels (Destruction/Repair)
21     AdaptiveWeightsAdjustment(S_best, S_cur, S_i)
22 return S_best, L_u
```

The repair operator $O_{rep}$ is selected (line 9) from among a set of repair operators (see Section 5.3). Next, it is used in $Construction(S_i, \mathcal{L}_{des}, \mathcal{L}_u, O_{rep})$ (line 10) to schedule all the unscheduled interventions ($\mathcal{L}_{des}$ and $\mathcal{L}_u$), when possible. Given an incumbent solution $S_i$, the order in which the $\mathcal{L}_{des}$ and $\mathcal{L}_u$ interventions are processed has an impact on the number of interventions that can be scheduled. This depends on free resources within intervals. The repair operators that we propose are based on how the interventions of $\mathcal{L}_{des}$ and $\mathcal{L}_u$ are sorted.

In order to escape from local optima, we allow the acceptance of a non-improving solution (line 11). The main issue to be faced is to avoid selecting a too low-quality solution, leading to a waste of processing time when exploring its neighborhood in the hope of obtaining a better quality solution. We chose to implement an adaptive acceptance strategy, which we detail in Section 5.5, based on the record-to-record travel algorithm proposed in Dueck (1993).

It is important to make the selection of good destruction/constructions op-

erators over the iterations possible in order to be able to schedule all the interventions. The adaptive mechanism that we implement is detailed in Section 5.4. The weights of the two roulette wheels are updated using $AdaptiveWeightsAdjustment(S_{best}, S_{cur}, S_i)$.

The number of iterations without any improvement $Iter_{max}$ can be fixed regardless of the instance or computed during the course of the ALNS algorithm according to some characteristics of the processed instance in order to adapt it to the difficulty of the instance during execution.

We chose to formulate $Iter_{max}$ as $\gamma^{\frac{|P \setminus \widetilde{P}|}{|R|}+1}.\frac{|\widetilde{P}|}{|R|}$ where $\gamma \in [1, 10]$. We recall that the set of scheduled interventions during the course of the algorithm is $\widetilde{P}$, the set of patients left unscheduled is $P \setminus \widetilde{P}$, and the set of resources is $R$. Therefore, $Iter_{max}$ mainly increases as $\frac{|\widetilde{P}|}{|R|}$ increases, the average number of scheduled interventions per resource. This allows us to spend more processing time when all interventions are scheduled to better improve the terms $O_{wt}$ and $O_{em}$ of the objective function. However, we also need to have a larger number of iterations when interventions are left unscheduled in order to schedule all interventions. The term $\gamma^{\frac{|P \setminus \widetilde{P}|}{|R|}+1}$ is equal to the value of parameter $\gamma$ when all interventions are scheduled since $|P \setminus \widetilde{P}|$ is zero because the set of interventions left unscheduled is empty, but it is larger when many interventions are left unscheduled.

The $\gamma$ parameter needs to be tuned to obtain a good trade-off between good quality solutions and processing times.

### *5.2. Destruction operators*

We investigated four destruction operators, two that are intervention/patient-oriented and two others are day-oriented. All these operators use the current solution $S_{cur}$ as input, and the intervention/patient operators remove a certain number of interventions while the day operators remove all the interventions scheduled in one day. Given a destroy operator $O_{des}$, the function $Destruction(S_{cur}, k, O_{des})$ (see Algorithm 2) returns $S_i$, the incumbent solution with the removed interventions, and $\mathcal{L}_{des}$ the list of removed interventions. The destruction operators we investigated are:

**RR** **R**andom **R**emoval (intervention/patient-oriented);

**WR** **W**orst **R**emoval (intervention/patient-oriented);

**RDR** **R**andom **D**ay **R**emoval (day-oriented);

**WDR** **W**orst **D**ay **R**emoval (day-oriented).

**R**andom **R**emoval removes $k$ surgical interventions at random from the solution and inserts them one after the other in $\mathcal{L}_{des}$. This can be done in constant time. **W**orst **R**emoval first assesses then sorts the interventions in decreasing order of cost; see equations (1a)-(1c). This can be done in $\mathcal{O}(|P| \cdot ln(|P|))$.

Next, the $k$ surgical interventions with the highest planning cost are removed. The planning cost of a patient $p$ is assessed by the cost difference, assuming the

withdrawal of the patient's intervention in the solution. This aims at decreasing the cost. **R**andom **D**ay **R**emoval selects one day in the planning horizon at random. Next, all the interventions scheduled for this day are removed from the solution. At most this can be done in $\mathcal{O}(|P|)$. **W**orst **D**ay **R**emoval first assesses all the scheduled interventions for each day, and then sorts the days in decreasing order of cost. Next, all the interventions for the day having the highest planning cost are removed from the solution. This can be done in $\mathcal{O}(|P| \cdot ln(|P|))$.

All these destroy operators aim at freeing resources, which allows the repair operators to schedule as many interventions as they can. **RR** and **RDR** are pure random destruction operators. They help diversify the search mechanism (Hemmati and Hvattum (2017)).

By focusing on days, **RDR** and **WDR** make it possible to reallocate resources within one day, while aiming at decreasing the cost, especially with regard to the third term $O_{em}$ of the objective function (1c) related to patients for whom the intervention should be planned early in the morning.

For the **W**orst **R**emoval, we assess the individual contribution for each scheduled intervention $p \in S_{cur}$. This allows us to remove the $k$ interventions with the highest cost. The **WDR** destroy operator also assesses the individual contributions. **WR** and **WDR** both work at the expense of more processing time but this make it possible to reduce the overall cost $Obj(S_i)$ using repair operators.

At each iteration, function $SelectDestructionOperator()$ (see Algorithm 2) selects one destruction operator $O_{des}$ that is chosen from among the four described above using an adaptive mechanism based on a roulette wheel selection algorithm (see Section 5.4).

The $Destruction(S_{cur}, k, O_{des})$ function requires a degree of destruction $k$, the number of interventions to be removed, that is used by **RR** and **WR**. Note that parameter $k = 1$ when a day-oriented operator is chosen since one day has to be selected. The degree of destruction $k$ affects diversification, and, consequently, the quality of solutions. It can be fixed, chosen at random from a range depending on the size of some characteristics of the instance, or gradually varied at run-time according to a strategy. We chose to implement an adaptive strategy for managing $k$ for **RR** and **WR** that uses the number of modes and the number of scheduled interventions that we obtain during the course of the ALNS algorithm.

In Algorithm 2, the value $k$ is chosen at random as $k \leftarrow rand(1, D_{max})$. Provided that an intervention/patient-oriented destroy operator has been chosen, $k$ is bounded by $D_{max}$. We define $D_{max}$ as the degree of diversification. This value is initialized to 3 and then incremented after each non-improving iteration as $D_{max} \leftarrow min(D_{max} + 1, D_{limit})$, when the $D_{limit}$ value is not attained. As soon as an improvement is found, we reset $D_{max}$ to 3 in order to entirely explore the neighborhood of the new best solution $S_{best}$.

The choice of the $D_{limit}$ value is important because it makes it possible to broaden the search around a solution to find a better solution. However, a too large value will possibly generate an unnecessary expenditure of processing time because a better solution could be found in another part of the search

space starting from a solution of lower quality that could be selected thanks to the acceptance mechanism (see Section 5.5).

We chose to formulate $D_{limit}$ as $\alpha.\frac{|\widetilde{P}|}{|M|}$ where $\widetilde{P}$ is the set of interventions (i.e., patients) already scheduled and $M$ is the possible modes that can be used. It is mainly proportional to the average number of scheduled interventions per mode that are used by the instance. Since $|\widetilde{P}| \leq |P|$ when a number of interventions are left unscheduled in solutions, this favors the adaptive acceptance strategy of low-quality solutions in order to find solutions where all interventions are scheduled. Since it increases as the number of scheduled interventions increases, it then makes it possible to better explore the neighborhood of solutions.

The size of $\widetilde{P}$ is larger than the size of $M$, and processing time increases as the number of removed interventions increases. We chose to select $\alpha \in [0, 1]$. The $\alpha$ parameter needs to be tuned to obtain good-quality solutions.

Over the iterations of the ALNS, the adaptive mechanism makes it possible to choose from among the four destruction operators to explore the neighborhood of a solution. The destruction operators shake up the current solution that makes it possible to explore the neighborhood of this solution.

### *5.3. Repair operators*

In the literature, the **B**est **I**nsertion **A**lgorithm is often used for solution repair. Overall, the idea is to insert items in an incumbent solution at a place chosen as the best one according to specific criteria (Amarouche et al., 2020; Ben-Said et al., 2019).

Based on the general principle, we proposed the dedicated algorithm as shown in Algorithm 3. Given an incumbent solution $S_i$, we compute the set that contains triplets of **s**urgeon, **r**oom and **t**ime **i**nterval that can be used to schedule an intervention. We obtain all the free $(s, r, \mathcal{I})$ relative to $S_i$ (modes and intervals). We denote this set as $\mathcal{S}_{sr(ti)}$, indexed by $sr(ti)$ for **s**urgeon, **r**oom and **t**ime **i**nterval. Initially computing $\mathcal{S}_{sr(ti)}$ takes $\mathcal{O}(H \cdot |P|)$ iterations, next $\mathcal{S}_{sr(ti)}$ is updated in constant time during the course of the ALNS algorithm.

The objective is to schedule as many interventions as possible while minimizing the cost. The algorithm uses the incumbent solution $S_i$ and an ordered/prioritized list of **i**nterventions **t**o **b**e **s**cheduled, denoted as $\mathcal{L}_{itbs}$. The list $\mathcal{L}_{itbs}$ contains the interventions left unscheduled of $\mathcal{L}_u$ and the interventions of $\mathcal{L}_{des}$ that have been removed by applying one of the four destruction operators. Given an initial $\mathcal{L}_{itbs}$ list, the BIA takes $\mathcal{O}(H \cdot |P|)$.

The way that interventions are ordered in $\mathcal{L}_{itbs}$ has an impact on how many interventions can be scheduled using the set $\mathcal{S}_{sr(ti)}$ of available resources within intervals. We propose and look into three different ways of building the order of interventions in $\mathcal{L}_{itbs}$. They are presented below.

Algorithm 3 first resets $\mathcal{L}_u$ to obtain the list of interventions left unscheduled for the forthcoming iteration of ALNS. Given a prioritized list $\mathcal{L}_{itbs}$, we attempt to insert the interventions one by one in the incumbent solution $S_i$. For an intervention $p$, we first determine the modes $m = (s, r) \in M_p$ and the associated intervals $\mathcal{I}$ from among the set of triplet $\mathcal{S}_{sr(ti)}$ that can be used to schedule

**Algorithm 3:** Best Insertion Patient

**Input** : $S_i$ incumbent solution at iteration $i$
$\mathcal{L}_{itbs}$ ordered/prioritized list of **i**nterventions **t**o **b**e **s**cheduled,
$\mathcal{S}_{sr(ti)}$ set of triplet (**s**urgeon, **r**oom, **t**ime **i**nterval) that can be used to schedule

**Output** : $S_i$ repaired incumbent solution at iteration $i$
$\mathcal{L}_u$ list of interventions left unscheduled

**Variables** : $\mathcal{S}_{p,sr(ti)c}$ for intervention $p$,
set of quadruplets (**s**urgeon, **r**oom, **t**ime **i**nterval, **c**ost)

$\mathcal{L}_u \leftarrow \emptyset$
**while** $(\mathcal{L}_{itbs} \neq nil)$ **do**
  // next intervention of ordered list of interventions to be scheduled
  $p \leftarrow \mathcal{L}_{itbs}.pop\_front()$
  $\mathcal{S}_{p,sr(ti)c} \leftarrow \emptyset$
  // select modes and time intervals for intervention $p$
  // assess cost
  **forall** $(s, r, \mathcal{I}) \in \mathcal{S}_{sr(ti)}$ **do**
    **if** $(m = (s, r) \in M_p) \wedge (\vartheta_p^m \leq size(\mathcal{I}))$ **then**
      $cost_p \leftarrow DiffCostAdd(S_i, p, s, r, \mathcal{I})$
      $\mathcal{S}_{p,sr(ti)c} \leftarrow \mathcal{S}_{p,sr(ti)c} \cup \{(s, r, \mathcal{I}, cost_p)\}$
  **if** $\mathcal{S}_{p,sr(ti)c} \neq \emptyset$ **then**
    $(p, s_{best}) \leftarrow SelectBest(\mathcal{S}_{p,sr(ti)c})$
    Schedule intervention $p$ in solution $S_i$ using mode $m$ and interval $\mathcal{I}$
    Update the set $\mathcal{S}_{sr(ti)}$
  **else**
    $\mathcal{L}_u.push\_back(p)$
**return** $S_i, \mathcal{L}_u$

this intervention. We then assess the related cost, denoted as $cost_p$, of inserting the intervention $p$ in $S_i$. We therefore obtain the set of quadruplet $\mathcal{S}_{p,sr(ti)c}$ (surgeon, room, interval and cost).

Provided that $\mathcal{S}_{p,sr(ti)c}$ is not empty, we select the best insertion quadruplet with minimum cost and we schedule intervention $p$ using the associated mode $m$ and interval $\mathcal{I}$. The set $\mathcal{S}_{sr(ti)}$ is then updated for the next iteration since the resource has been used. If the set $\mathcal{S}_{p,sr(ti)c}$ is empty, the intervention $p$ cannot be scheduled and is added to the list $\mathcal{L}_u$.

The adapted insertion algorithm and the ordered/prioritized list $\mathcal{L}_{itbs}$ together play the role of a repair operator. We can have different solutions using different priority lists. Given the two lists $\mathcal{L}_u$ and $\mathcal{L}_{des}$, we first append list $\mathcal{L}_{des}$ behind list $\mathcal{L}_u$ to obtain the initial list $\mathcal{L}_{itbs}$. We then propose three ways of ordering/prioritizing interventions in $\mathcal{L}_{itbs}$:

**UIF** **U**nscheduled **I**nterventions **F**irst;

**NRM** sorted in ascending order of the **N**umber of **R**equested **M**odes;

**AC** sorted in ascending order of an **A**ggregation of **C**riteria.

Computing the list $\mathcal{L}_{itbs}$ takes at most $\mathcal{O}(|P| \cdot ln(|P|))$.

The first repair operator **UIF** aims at giving priority to the unscheduled interventions remaining in $\mathcal{L}_u$. The idea is to free resources by applying a destruction operator to first schedule the interventions of $\mathcal{L}_u$, we then schedule the interventions of $\mathcal{L}_{des}$.

Overall, the idea behind the **NRM** repair operator is to first schedule the interventions with the least number of modes, assuming that we will have more possibilities to schedule the others as things progress when we schedule the interventions.

More generally, it may be interesting to consider all the elements related to the scheduling of a patient's intervention. We must not lose sight of the fact that the medical constraints of hard due date $H_p$ and of due date $D_p$ should also be taken into consideration. The modes for scheduling the intervention of a patient as well as the resources may have an impact.

We propose to investigate a strategy for the third repair operator based on the aggregation of the criteria related to the patient's level of urgency ($H_p$ and $D_p$) and those related to modes and resources. Let $n_{mode}$ be the number of different modes that can be used for a patient and let $n_{res}$ be the number of different resources that we count over the different modes. We first compute $H_p \,.\, D_p \,.\, n_{res} \,.\, n_{mode}$, and then sort $\mathcal{L}_{itbs}$ in ascending order of these values. A low value indicates that the patient is urgent, that a resource is critical, and that there are few modes for the patient's intervention.

The idea behind the **AC** repair operator is to first schedule the patients' interventions with low values, assuming that we will have more possibilities to schedule the others as things progress when we schedule the interventions.

Over the iterations of the ALNS, the adaptive mechanism makes it possible to choose from among the three repair operators to explore the neighborhood of a solution.

### *5.4. Adaptive mechanism for destruction/repair*

The idea is to promote future choices of successful operators by modifying the weights used in a roulette wheel algorithm that selects operators. The weights for the roulette wheel used for destruction operators and the weights for the other roulette wheel used for repair operators are updated using the *AdaptiveWeightsAdjustment* procedure according to the obtained performance.

The proposed ALNS algorithm uses four destruction and three repair operators. Operator success may vary depending on the problem instance. The adjustment of the roulette wheel weights is necessary to increase the probability that efficient operators are used more often than less efficient operators. Dynamic adjustment is the only way to ensure a permanent re-evaluation of operators' weights.

Two approaches can be used for updating the weights: at each iteration or periodically after a certain number of iterations. The advantage of the first one is that weights are up-to-date at all times but at the expense of processing time since the weights and the operator probabilities for the next iteration have to be calculated at each iteration. On the contrary, by periodically updating the weights after a certain number of $p_u$ iterations, we save processing time. However, this implies determining a good value for $p_u$. A large value saves processing time but does not allow the adaptive mechanism to fully play its role, which can result in poor-quality solutions. To save processing time, we

chose the second approach. Hence, the tuning of $p_u$ is needed to obtain a good trade-off between saved computing time and good-quality solutions.

The adaptive mechanism is the same for destruction or repair operator selection. We denote $O$ as either a destruction or a repair operator. The number of times the operator $O$ has been used during $p_u$ iterations is denoted as $u(O)$. The success of operator $O$ is denoted as $s(O)$, set to zero at the beginning of the period of $p_u$ iterations. Given the incumbent solution $S_i$, the current solution $S_{cur}$, and the best solution found so far $S_{best}$, the value of $s(O)$ is increased at each iteration as:

$$s(O) + \sigma_i \quad \text{where } \sigma_i \text{ is either} \quad \begin{cases} \sigma_1 & \text{if} \quad Obj(S_i) \leq Obj(S_{best}) \\ \sigma_2 & \text{if} \quad Obj(S_i) \leq Obj(S_{cur}) \\ \sigma_3 & \text{otherwise} \end{cases}$$

In the first case, the operator $O$ improves or equals the best solution found so far, while in the second case it improves or equals the current solution. In the last case, we reward the operator when a new solution is worse but accepted according to the acceptance strategy (see Section 5.5). We have $\sigma_1 > \sigma_2 > \sigma_3$. This promotes the operator as the relative or absolute solution quality increases. These parameters need to be tuned because they have an impact on the solution quality. We chose to formulate $\sigma_2$ as $\sigma_2 = \theta.\sigma_1$ and $\sigma_3$ as $\sigma_3 = \phi.\sigma_2$ where $\theta, \phi \in [0, 1]$. Provided that an initial value is set for $\sigma_1$, the parameter tuning of $\sigma_1, \sigma_2, \sigma_3$ can be performed by varying the couple of parameters $(\theta, \phi)$.

The weights for the forthcoming iterations are calculated as:

$$w(O) = \begin{cases} (1-l)w(O) + l\frac{s(O)}{u(O)}, & \text{if } u(O) > 0 \\ (1-l)w(O), & \text{if } u(O) = 0 \end{cases}$$

The reaction factor $l \in [0, 1]$ is a parameter that controls the influence of the recent success of an operator on its weight.

The four destruction and the three repair operators together make large changes possible from one solution to its neighbors, which may prevent the search process from being stuck in local optima. However, since operator success may vary depending on the problem instance, the adaptive mechanism plays an important role by rewarding efficient operators throughout iterations. To make the adaptive mechanism efficient, the parameters presented above need to be tuned.

### *5.5. Adaptive acceptance strategy*

We chose to implement an adaptive acceptance strategy based on the record-to-record travel algorithm. Originally proposed by Dueck (1993), it is a deterministic variant of simulated annealing. A low-quality solution is accepted based on the difference between the incumbent solution cost $Obj(S_i)$ and the best solution cost $Obj(S_{best})$ relative to a value denoted as record-deviation, which is a threshold-based strategy. This straightforward but efficient mechanism has

the advantage that it depends only on one parameter, which is the value of the record-deviation, denoted here as $Dev$ (see Dueck (1993)).

The record-deviation parameter $Dev$ plays a central role in controlling the acceptance of low-quality solutions. In the general scheme of the algorithm presented in Dueck (1993), the record-deviation parameter is fixed and originally used as $Dev \leq Obj(S_i) - Obj(S_{best})$. We chose to implement it relative to a percentage as $Obj(S_i) \leq (1+Dev).Obj(S_{best})$ because the unscheduled interventions (postponed from a surgical standpoint) strongly contribute to the overall cost of the objective function (1a)-(1c). It is preferable to use this percentage rather than an absolute difference value for comparison purposes between instances for which all interventions are scheduled and those for which all interventions are not scheduled.

However, the larger this parameter is, the more we unnecessarily explore low-quality solutions because the record-to-record-based mechanism is activated too early.

To have the opportunity to first explore the neighborhood of a current solution using the destroy/repair operators we propose an adaptive acceptance strategy that we manage in procedure $Accept(S_{best}, S_i)$ (see Algorithm 2). Thus, the value of $Dev$ is not fixed, it can increase or decrease throughout the search space exploration process.

However, the increase of $Dev$ may be at the expense of the adaptive diversification mechanism that also increases $D_{max}$, the maximum number of interventions that can be removed from $S_i$, when a lower quality solution is met. In Algorithm 2, we update $D_{max} \leftarrow min(D_{max}+1, D_{limit})$ when $Obj(S_i) \geq Obj(S_{best})$ for some destroy operators. A good trade-off between these two mechanisms must be found to prioritize local searches.

In procedure $Accept(S_{best}, S_i)$, we implement an adaptive mechanism that increases the value of $Dev$ by a constant value $\Delta_{Dev}$ when no improvement is met throughout a defined number of iterations $Iter_D$. The record-deviation parameter is equal to zero at the beginning ($Dev = 0$). When the defined number of iterations $Iter_D$ without improvement has been performed, $Dev$ is increased by the record-deviation increment $\Delta_{Dev}$ to permit the diversification mechanism to play its role. Low-quality solutions are more often accepted, which allows the search to jump out from local optima. The $Dev$ value increases by $\Delta_{Dev}$ as long as destroy/repair operators fail to find a better solution. We reset the record-deviation parameter $Dev$ to zero if a local or global improvement occurs.

Therefore, the two parameters to be tuned to make this adaptive acceptance strategy efficient are $\Delta_{Dev}$, the record-deviation increment, and, $Iter_D$ the number of iterations without improvement that induces the incrementation of $Dev$.

## 6. Computational experiments

In our experiments, our objectives were: (i) to show the effectiveness of the 2PSC-EM heuristic compared to the literature results; (ii) to provide a summary

and insights into how to conduct the tuning of the ALNS so that it can be used in other situations where planning of elective patients has to be done over a period of several months; (iii) to evaluate the effectiveness of the destruction/repair operators and the adaptive mechanism for destruction/repair and to discuss the results to show whether a component contributes to obtaining good solutions by also considering the processing time; and (iv) to assess the quality of the solutions computed by the ALNS on benchmark instances compared to the literature results. Tests were done using C++ and Standard Template Library (STL), compiled with GCC under Linux, on a machine with an Intel(R) Xeon(R) Gold 6138 CPU @ 2.00 GHz.

### *6.1. Benchmark overview and experimental protocol*

We tested the 2PSC-EM heuristic and the ALNS meta-heuristic on the admission set of the benchmark proposed by Riise et al. (2016). The authors designed an instance generator based on the real resources of a surgery department (Bærum Sykehus, Norway). For confidentiality reasons, the patients are generated. The test data are available on-line in XML format (SINTEF, 2013).

In this paper we focus on building the scheduling of surgical interventions, i.e. admission planning, for which it is of interest for the hospital organization to know whether it is possible to schedule the interventions of elective patients regardless of whether they are outpatients or inpatients. Each of the ten instances of the benchmark test consists of 728 patients to be scheduled over a planning horizon of between 65 and 71 days (about 3 months). In all instances, there are seven surgeons and four operating rooms.

In Riise et al. (2016), the authors decided to run the ACI algorithm ten times for each instance and a time limit of 600 s has been chosen for each run. They reported the minimum, maximum and average solution values at the chosen time limit of 600 s. In Riise et al. (2016), the best solution value, denoted as UB, is also reported for each instance. This UB value corresponds to the best value obtained on all the runs performed by the authors. The processing times for obtaining the UB values are not reported in the paper. The authors indicated that they are obtained using considerably longer running times than the 600 s time limit.

### *6.2. Comparison of 2PSC-EM heuristic with the literature*

In Table 2, for each instance under the *ACI* heading, the *UB* column shows the best known value, or upper bound, the *Min* column shows the minimum value obtained by ACI (see Riise et al. (2016)), the *Mi2UB* column shows the percentage $(UB - Min)/UB$, and the *t(s)* column shows the processing time. Given that the weight $W_{un}$ is set to 1,000 (see term (1a)), values larger than a thousand indicate that some interventions are left unscheduled. The results that are better than UB are shown in bold print.

Since our algorithm is also a randomized one like ACI, we chose to initially perform 100 runs to show whether the heuristic can obtain better results. A run of the 2PSC-EM heuristic takes about ten seconds. Under the *2PSC-EM*

*(100)* heading, we show the minimum value we obtain and the percentage $(UB-Min)/UB$ in the *Min* and *Mi2UB* columns, respectively.

In eight out of ten instances the 2PSC-EM heuristic obtains better results. For the eight instances for which all the interventions are scheduled using the ACI approach and the proposed 2PSC-EM heuristic, we improve UB up to 16% and 9% on average.

| | ACI | | | | 2PSC-EM (100) | | | 2PSC-EM (3) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | UB | Min | Mi2UB% | t(s) | Min | Mi2UB% | t(s) | Min | Mi2UB% | t(s) |
| a_01 | 21.58 | 21.85 | -1.27 | 600 | **21.15** | 1.97 | 1319 | 22.00 | -1.94 | 42.0 |
| a_02 | 22.83 | 23.25 | -1.83 | 600 | **19.72** | 13.63 | 1335 | **20.32** | 11.01 | 40.0 |
| a_03 | 24.52 | 24.58 | -0.26 | 600 | **22.55** | 8.02 | 918 | **22.66** | 7.81 | 31.2 |
| a_04 | 21.97 | 22.28 | -1.40 | 600 | **20.43** | 7.02 | 958 | **20.62** | 6.16 | 29.7 |
| a_05 | 5113.88 | 6341.84 | -24.01 | 600 | 11025.10 | -115.59 | 934 | 13025.30 | -154.71 | 28.2 |
| a_06 | 23.55 | 23.55 | 0.03 | 600 | **22.15** | 5.97 | 991 | **22.68** | 3.72 | 31.6 |
| a_07 | 22.73 | 23.25 | -2.28 | 600 | **21.98** | 3.31 | 1026 | **22.44** | 1.28 | 30.0 |
| a_08 | 2086.32 | 3148.49 | -50.91 | 600 | 5024.88 | -140.85 | 797 | 10024.70 | -380.50 | 23.9 |
| a_09 | 24.83 | 24.99 | -0.64 | 600 | **20.75** | 16.43 | 926 | **21.23** | 14.50 | 29.2 |
| a_10 | 24.48 | 24.74 | -1.08 | 600 | **20.61** | 15.83 | 935 | **20.92** | 14.53 | 25.9 |

Table 2: 2PSC-EM heuristic, 100 runs and three runs, compared to ACI.

Except for the a_05 and the a_08 instances for which neither the ACI nor the 2PSC-EM heuristic succeeded in scheduling all the interventions, the 2PSC-EM heuristic that we propose provides better results. These preliminary results using the 2PSC-EM heuristic show that better results are achieved. However, in some cases, some interventions are still left unscheduled. A more effective approach is to be investigated with the aim of scheduling all the interventions, when possible.

The 2PSC-EM heuristic can potentially provide good initial solutions to the ALNS approach that we propose to study here. However, we cannot spend too much processing time to obtain these solutions. We therefore perform another series of experiments where we limit the number of runs to show whether good initial solutions can be obtained using the heuristic using a limited amount of time, while not significantly worsening the quality of the solutions. Since the heuristic is a randomized method, we performed several experiments of k runs. We experimentally observed that $k = 3$ runs seems to be a good trade-off between solution quality and processing time.

In Table 2 under the heading *2PSC-EM (3)*, we show the best results obtained using the 2PSC-EM heuristic when performing three runs of the heuristic ten times. In seven out of ten instances, the 2PSC-EM heuristic still obtains better results. A small degradation is observed for the instance a_01 compared to UB. For the eight instances for which all the interventions can be scheduled, the proposed 2PSC-EM heuristic still improves UB up to 14% and 7.1% on average. As can be shown, slightly better results can be achieved using 100 runs of the 2PSC-EM heuristic compared to three runs, but it takes a while to obtain them. We experimentally observed that three runs are sufficient to obtain adequate solutions within a short processing time.

We propose the ALNS approach with the aim of scheduling all the interventions and improving the results of the 2PSC-EM heuristic. As initial solutions of the ALNS approach, we use the best solution computed using three runs of the 2PSC-EM heuristic below.

*6.3. Parameter tuning*

We present the experiments we carried out to find the parameter values that were used to test our ALNS approach. As described in Section 5.1, the ALNS that we propose has four sets of parameters:

- $D_{limit}$, the maximum number of patients removed by the destroy operators;
- $\Delta_{Dev}$ and $Iter_D$, the record-deviation increment and the number of iterations without improvement before increasing the deviation;
- $\sigma_1, \sigma_2, \sigma_3$, $p_u$, and $l$, the first three are the added scores after using a destroy or repair operator, $p_u$ is the number of iterations before adjusting the weights of operators, and $l$ is the reaction factor;
- $Iter_{max}$, the maximum number of iterations without improvements.

Some of these parameters are computed during the course of the ALNS algorithm according to some characteristics of the instance being processed or are relative to other parameters. For these parameters, the first column in Table 3 shows the parameters as presented in Section 5.1. The second column describes how they are computed and the related parameters that need to be tuned.

| | Computed as, and parameters to be tuned |
|---|---|
| $D_{limit}$ | $\alpha.\frac{\lvert\tilde{P}\rvert}{\lvert M\rvert}$, parameter $\alpha$ to be tuned |
| $\sigma_1, \sigma_2, \sigma_3$ | $\sigma_1$ fixed, next $\sigma_2 = \theta.\sigma_1$ and $\sigma_3 = \phi.\sigma_2$ parameters $(\theta, \phi)$ to be tuned |
| $Iter_D$ | $\delta.Iter_{max}$, parameter $\delta$ to be tuned |
| $Iter_{max}$ | $\gamma^{\frac{\lvert P\setminus\tilde{P}\rvert}{\lvert R\rvert}+1}.\frac{\lvert\tilde{P}\rvert}{\lvert R\rvert}$, parameter $\gamma$ to be tuned |

Table 3: Parameters $\alpha$, $\theta$, $\phi$, $\delta$ and $\gamma$ to be tuned.

To calibrate all the parameters, we carried out preliminary experiments on a subset of five instances that are randomly selected from the eight benchmark instances for which all the interventions can be scheduled. Since ALNS is a randomized search method, the experiments were repeated five times with a different random seed. To tune the parameters, we evaluate the results using the Relative Percentage Error (RPE). We define the RPE as follows: $RPE = 100 \times \frac{Z_{best} - Z_{max}}{Z_{best}}$ where $Z_{best}$ denotes the best result we achieved over all the runs we performed for an instance, and $Z_{max}$ denotes the best result we obtained among the five performed runs. Next, the RPE values are averaged for the five

instances chosen at random, and this average is shown as a percentage in the figures we present and that we comment on below.

We start by tuning the first $D_{limit}$, and we then tune each parameter, one after the other, considering the four sets of parameters presented. The initial values we use to first tune $D_{limit}$ are $\Delta_{Dev} = 0.2$, $\delta = 0.2$, $\sigma_1 = 50$, $\theta = 0.3$, $\phi = 0.1$, $p_u = 60$, $l = 0.5$ and $\gamma = 10$. Then, to tune a parameter, we retain the best setting of the other parameters found before proceeding to tune it.

**The maximum number of patients to be removed $D_{limit}$**

We performed the tuning experiments by varying $\alpha$ from 0.1 to 1.0 with a step of 0.05. In Figure 1, we show that when $\alpha$ is lower than 0.2, the number of interventions removed from the solution is too small, so the neighborhood exploration is insufficient. When $\alpha$ is larger than 0.35, the number of interventions removed from a current solution increases. The diversification increases so the quality of the solutions is worsened, as can be seen by the average RPE values.

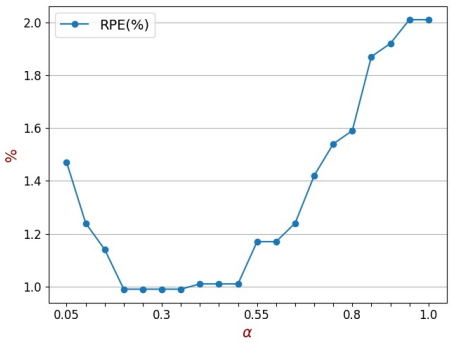


Figure 1: Tuning of $D_{limit}$, effect of $\alpha$ on RPE.

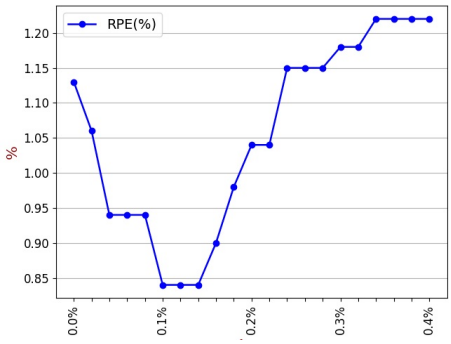


Figure 2: Tuning of $\Delta_{Dev}$, effect of on RPE.

As could be expected, the processing time increases with the $\alpha$ value, so we do not show a curve of CPU time vs $\alpha$. The larger the values of $\alpha$ are, the larger the processing times are, which is ineffective. We chose $\alpha = 0.3$, which is a good trade-off between solution quality and processing time.

**Record-to-record-based adaptive acceptance strategy; tuning of $\Delta_{Dev}$ and $Iter_D$**

To calibrate the value of $\Delta_{Dev}$ we varied it between 0.0% and 2.5% with increments of 0.02%. In Figure 2, we show that when $\Delta_{Dev}$ varies between zero and 0.1%, the RPE decreases. We then reach a plateau of the curve for three values that give the smallest value of RPE. Next, the average RPE values increase as $\Delta_{Dev}$ increases. We observe this same phenomenon for values of $\Delta_{Dev}$ larger than 0.4%; consequently, values larger than 0.4% have not been

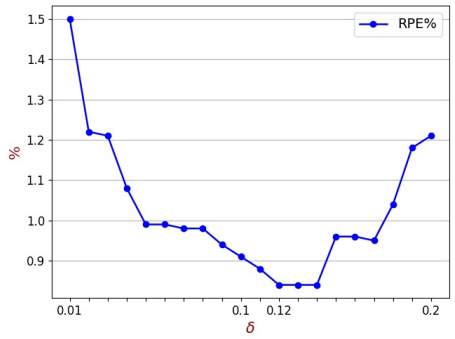


Figure 3: Tuning of $Iter_D$, effect of $\delta$ on RPE.

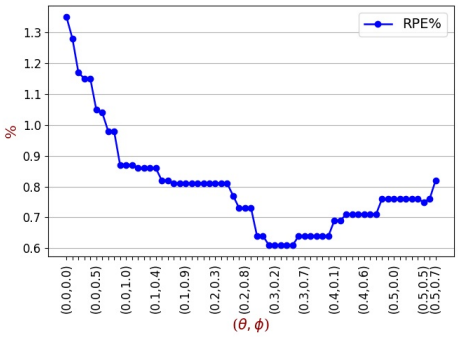


Figure 4: Tuning of $\sigma_2$ and $\sigma_3$ relative to $\sigma_1$, effect of $\theta$ and $\phi$ on RPE.

shown in Figure 2. We fix the value of $\Delta_{Dev} = 0.1\%$ for the record-deviation increment. It is the smallest value for which the average RPE value is minimum.

We then proceed to the tuning of $\delta$ for $Iter_D$, the number of iterations without improvement before the deviation increase. In Figure 3, three good values of $\delta$ can be chosen; the first is 0.12. If $Iter_D$ is small, the allowed deviation rapidly increases and the algorithm accepts poor quality solutions. On the other hand, if $Iter_D$ is large, the deviation does not increase as much and the algorithm will be trapped in a local optimum. We chose $\delta = 0.12$.

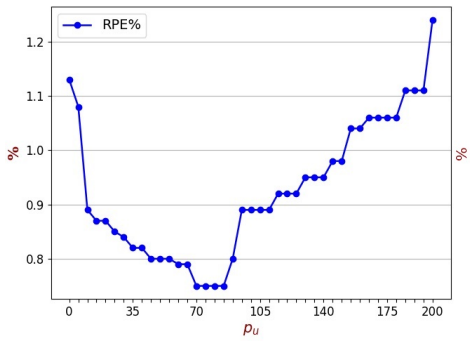


Figure 5: Effect of $p_u$ on RPE.

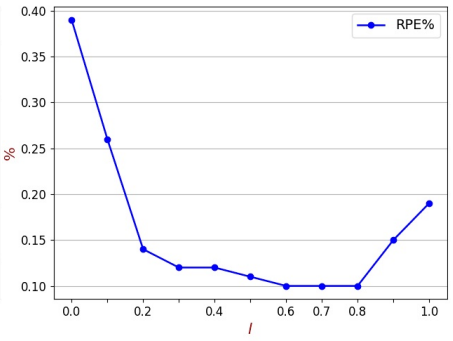


Figure 6: Effect of $l$ on RPE.

**Adaptive weight adjustment parameters; tuning of $\sigma_1, \sigma_2, \sigma_3, p_u$ and $l$**

We first tune the weight adjustment parameters $\sigma_1, \sigma_2, \sigma_3$, the scores added after the use of a destruction or repair operator. Given that $\sigma_1 > \sigma_2 > \sigma_3$ according to the usual reward of operators relative to the quality of the incumbent

solution $S_i$, the current solution $S_{cur}$, and the best solution found so far, $S_{best}$, we chose to first set $\sigma_1$ to 50 and we compute $\sigma_2 = \theta.\sigma 1$ and $\sigma_3 = \phi.\sigma_2$ such that $\theta, \phi \in [0, 1]$.

Figure 4 shows the beginning of the curve that we obtain to better focus on the interesting values. We chose $\theta = 0.3$ and $\phi = 0.2$, the first value giving the best performance relative to the average RPE.

To tune $p_u$, the number of iterations before adjusting the weights of operators, we vary it between 0 and 200 iterations with a step of 5. Figure 5 shows that the RPE is better when $p_u$ is between 70 and 85, and we chose to set $p_u$ to 70.

To find the best value of $l$, the reaction factor that controls the influence of the recent success of an operator on its weight, we vary it between 0 and 1 with an increment of 0.1. The average $RPE$ values shown in Figure 6 reveal that good solutions are achieved when $l$ is between 0.6 and 0.8. The previous iteration seems to have a little influence on the weight calculation. We chose to fix $l$ at 0.7.

**Maximum number of iterations without improvement; tuning of $Iter_{max}$**

Finally, we proceed to the tuning of $Iter_{max}$, the maximum number of iterations without improvement. The processing time needed to find a good quality solution varies according to the instances and this parameter has an important impact. We chose to formulate $Iter_{max}$ as $\gamma^{\frac{|P/\widetilde{P}|}{|R|}+1} \cdot \frac{|\widetilde{P}|}{|R|}$ where $\alpha \in [1, 10]$, $|\widetilde{P}|$ is the number of scheduled interventions, $|P/\widetilde{P}|$ is the number of unscheduled ones and $|R|$ is the number of resources.

As would be expected, the average RPE decreases as $\gamma$ increases, and, the processing time increases with the $\gamma$ value. Therefore, for the sake of compactness, we do not show curves for the $\gamma$ tuning. For RPE, the shape of the curve has an asymptotic behavior like a $1/x$ one. The larger the values of $\gamma$ are, the greater the processing times are, which is ineffective. We chose $\gamma = 11$, which is a good trade-off between solution quality and processing time and that generally remains lower than 250 s.

The final calibration results are $\alpha = 0.3$, $\Delta_{Dev} = 0.1\%$, $\delta = 0.1$, $\sigma_1 = 50$, $\theta = 0.5$, $\phi = 0.2$, $p_u = 70$, $l = 0.7$ and $\gamma = 11$. The values of the parameters are used in the sequel for our experiments on all the benchmark instances. They were chosen to obtain a good trade-off between solution quality and processing time.

### *6.4. Evaluation of the ALNS components*

We evaluate here the effectiveness of the destruction/repair operators and of the adaptive mechanism for destruction/repair by disabling each of them one at a time. This allows us to discuss the results to show whether a component contributes to obtaining good solutions by also considering the processing time.

In Table 4, we show the results for the four destruction operators (see Section 5.2). Under each heading, we show the minimum values (*Min* column) and the average processing times in seconds (*t(s)* column) that we obtain within 10 runs.

For comparison purposes, we tabulate under the *ALNS* heading the values that we obtain running the ALNS with all of components activated (also in Table 5).

| | ALNS | | No RR | | No WR | | No RDR | | No WDR | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Min | t(s) | Min | t(s) | Min | t(s) | Min | t(s) | Min | t(s) |
| a_01 | **20.64** | 116.25 | 20.89 | 85.09 | 20.69 | 193.74 | 20.65 | 168.87 | 20.69 | 183.92 |
| a_02 | **19.41** | 119.86 | 19.59 | 109.17 | 19.50 | 132.04 | 19.44 | 120.89 | 19.50 | 159.37 |
| a_03 | **22.13** | 117.25 | 22.58 | 120.99 | 22.21 | 196.98 | 22.22 | 190.22 | 22.18 | 226.77 |
| a_04 | **20.37** | 129.81 | 20.49 | 94.58 | 20.46 | 139.49 | **20.37** | 135.38 | 20.43 | 264.55 |
| a_05 | **23.77** | 1109.92 | 6027.92 | 301.24 | 24.00 | 4045.17 | 24.05 | 4624.39 | 24.58 | 3268.68 |
| a_06 | **22.30** | 72.59 | 22.44 | 53.26 | 22.34 | 81.75 | 22.39 | 161.33 | 22.39 | 100.86 |
| a_07 | **21.44** | 93.54 | 21.56 | 107.37 | 21.50 | 113.31 | **21.44** | 123.28 | 21.45 | 182.66 |
| a_08 | **22.59** | 548.21 | 3025.79 | 147.91 | 22.61 | 1980.98 | 22.85 | 1655.09 | 22.74 | 1560.71 |
| a_09 | **20.52** | 77.80 | 20.57 | 69.03 | 20.55 | 95.83 | 20.57 | 118.73 | **20.52** | 137.31 |
| a_10 | **20.33** | 78.33 | 20.36 | 122.60 | 20.35 | 117.41 | 21.82 | 109.47 | 20.34 | 131.32 |

Table 4: Impact of destruction operators RR, WR, RDR and WDR.

The ALNS without **R**andom **R**emoval (*No RR* column) obtains results that are below those of ALNS using all of the components. The **RR** destruction operator is necessary, and the patients' interventions chosen at random make it possible to avoid being stuck in a local optimum. This is especially the case for the instances a_05 and a_08 for which we cannot schedule all the interventions if **RR** is disabled. The processing times are shorter but we experimentally observe that the ALNS with no **RR** stops because we perform $Iter_{max}$ iterations without any improvement.

The ALNS without **W**orst **R**emoval (*No WR* column) obtains results that are below those of ALNS using all of the components. For all instances, all the interventions can be scheduled but at the expense of larger processing times. The **WR** destruction operator is necessary. It is a performance-oriented destruction operator that aims at minimizing the objective function, which is necessary to obtain better overall performances.

The **R**andom **D**ay **R**emoval (RDR) and **W**orst **D**ay **R**emoval (WDR) are day oriented destruction operators that make it possible to reallocate resources within one day. The ALNS without **R**andom **D**ay **R**emoval (*No RDR* column) obtains results that are below those of ALNS using all of the components except for instances a_04 and a_07 for which we attain the same best results. The **RDR** destruction operator is necessary, although for all instances for which all the interventions can be scheduled we, observe that it is at the expense of larger processing times.

The ALNS without **W**orst **D**ay **R**emoval (*No WDR* column) obtains results that are below those of ALNS using all components except for instance a_09 for which we attain the same best result. For all instances, all the interventions can be scheduled but at the expense of larger processing times, so **WDR** is necessary too.

To summarize, the four destruction operators are necessary. They are beneficial for exploring the neighborhood of a solution either by obtaining better results or by shortening the processing time.

| | ALNS | | No UIF | | No NRM | | No AC | | No Adaptive | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Min | t(s) | Min | t(s) | Min | t(s) | Min | t(s) | Min | t(s) |
| a_01 | **20.64** | 116.25 | 20.70 | 226.27 | 20.76 | 141.14 | 20.75 | 149.44 | 20.85 | 238.43 |
| a_02 | **19.41** | 119.86 | 19.42 | 259.44 | 19.49 | 117.86 | 19.45 | 120.27 | 19.59 | 240.48 |
| a_03 | **22.13** | 117.25 | 22.43 | 363.59 | 22.18 | 256.03 | 22.21 | 284.99 | 22.27 | 216.09 |
| a_04 | **20.37** | 129.81 | 20.41 | 187.91 | 20.38 | 144.80 | 20.43 | 161.11 | 20.37 | 532.06 |
| a_05 | **23.77** | 1109.92 | 24.35 | 3777.95 | 24.28 | 3680.23 | 24.55 | 4328.10 | 28.37 | 5999.34 |
| a_06 | **22.30** | 72.59 | 22.39 | 188.19 | 22.39 | 71.11 | 22.39 | 119.32 | 22.42 | 92.27 |
| a_07 | **21.44** | 93.54 | 21.52 | 188.27 | 21.52 | 133.41 | 21.52 | 182.70 | 21.57 | 163.06 |
| a_08 | **22.59** | 548.21 | 22.73 | 1681.28 | 23.34 | 1733.85 | 22.83 | 1739.60 | 25.61 | 2310.89 |
| a_09 | **20.52** | 77.80 | 20.56 | 171.90 | 20.60 | 79.87 | 20.53 | 125.69 | 20.55 | 230.76 |
| a_10 | **20.33** | 78.33 | 20.34 | 188.63 | 20.35 | 115.90 | 20.37 | 104.35 | 20.35 | 236.90 |

Table 5: Impact of repair operators UIF, NRM and AC, and, impact of adaptive mechanism.

In Table 5, we show the results for the three repair operators (see Section 5.3) and for the adaptive mechanism (see Section 5.4) that manages the choice of destruction and repair operators.

Overall, for the three repair operators, **U**nscheduled **I**nterventions **F**irst (UIF), **N**umber of **R**equested **M**odes (NRM) and **A**ggregation of **C**riteria (AC), we first observe that none succeeded in attaining the results of the ALNS approach using all of the components, and, second, that processing times are generally larger. This is especially the case for the instances a_05 and a_08.

To summarize, the three repair operators are necessary, and are beneficial for exploring the neighborhood of a solution either by obtaining better results or by shortening the processing time.

The last experiment was conducted to highlight how efficient the adaptive mechanism is. We disabled this mechanism by fixing the same probability for every destruction operator (1/4) and the same probability for every repair operator (1/3).

As can be seen in *No Adaptive* column, we obtain results that are below those of ALNS that uses this component, and the processing times are greater. This is particularly evident for instances a_05 and a_08 for which without this adaptive mechanism we obtain the worst processing times of these series of experiments. The adaptive mechanism that manages the choice of destruction and repair operators during the course of the ALNS is necessary to achieve good results within good processing time for the elective patient admission planning problem that we address here.

### *6.5. Comparison of ALNS with the literature*

In Table 6, for each instance under the *ACI* heading, we show the results obtained by ACI (see Riise et al. (2016)). Column *UB* shows the best known value, or upper bound, and columns *Min*, *Max*, *Avg* show the minimum value, the maximum value and the average value. Under the *ALNS* heading, we also show the *Min*, *Max* and *Avg* values that we obtained. Column *t(s)* shows the average processing times in seconds. We show the results that are better than UB in bold print.

| | ACI | | | | | | | ALNS | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UB | Min | Max | Avg | Mi2Av% | Mi2UB% | t(s) | Min | Max | Avg | Mi2Av% | Mi2UB% | Mi2Mi% | t(s) |
| a_01 * | 21.58 | 21.85 | 22.23 | 22.06 | 0.95 | -1.25 | 600 | **20.64** | **20.94** | **20.77** | 0.63 | 4.36 | 5.54 | 116.25 |
| a_02 * | 22.83 | 23.25 | 23.56 | 23.4 | 0.64 | -1.84 | 600 | **19.41** | **19.60** | **19.53** | 0.61 | 14.98 | 16.52 | 119.86 |
| a_03 * | 24.52 | 24.58 | 24.88 | 24.74 | 0.65 | -0.24 | 600 | **22.13** | **22.49** | **22.28** | 0.67 | 9.75 | 9.97 | 117.25 |
| a_04 * | 21.97 | 22.28 | 22.66 | 22.48 | 0.89 | -1.41 | 600 | **20.37** | **20.48** | **20.45** | 0.39 | 7.28 | 8.57 | 129.81 |
| a_05 | 5113.88 | 6341.85 | 9397.78 | 7987.31 | 20.60 | -24.01 | 600 | **23.77** | **4024.46** | **1425.0** | 98.33 | 99.54 | 99.63 | 1109.92 |
| a_06 * | 23.56 | 23.56 | 24.03 | 23.82 | 1.09 | 0.00 | 600 | **22.30** | **22.44** | **22.41** | 0.49 | 5.35 | 5.35 | 72.59 |
| a_07 * | 22.73 | 23.26 | 23.57 | 23.39 | 0.56 | -2.33 | 600 | **21.44** | **21.62** | **21.54** | 0.46 | 5.68 | 7.82 | 93.54 |
| a_08 | 2086.32 | 3148.49 | 4244.57 | 3407.31 | 7.60 | -50.91 | 600 | **22.59** | 4023.4 | **423.21** | 94.66 | 98.92 | 99.28 | 548.21 |
| a_09 * | 24.83 | 24.99 | 25.98 | 25.37 | 1.50 | -0.64 | 600 | **20.52** | **20.64** | **20.60** | 0.39 | 17.36 | 17.89 | 77.80 |
| a_10 * | 24.47 | 24.75 | 25.1 | 24.9 | 0.60 | -1.14 | 600 | **20.33** | **20.42** | **20.36** | 0.15 | 16.92 | 17.86 | 78.33 |
| for '*' | | | | | 0.86 | -1.11 | | | | | 0.47 | 10.21 | 11.19 | 100.67 |

Table 6: ALNS compared to ACI

For comparison purposes, we use $(Min - Avg)/Avg$ denoted in column headings as *Mi2Av* and $(UB - Min)/UB$ denoted in column headings as *Mi2UB*. The *Mi2Mi* column shows the $(Min(ACI) - Min(ALNS))/Min(ACI)$ values. The first allows us to show how close to the minimum the average results of ACI or of ALNS (for ten runs) are for each instance, whereas the second shows how far the results are from the best known UB value. The first is only really significant for instances for which all interventions can be scheduled. As a matter of fact, the objective function (see Equation (1a)-(1c)) is a weighted sum of normalized terms. The weight $W_{un}$ used for the term that assesses the interventions left unscheduled is the largest compared to the others. This clearly shows that unscheduled interventions are a key issue of great concern. It is not fair to compare the results of instances for which all interventions are scheduled to those for which some are left unscheduled because the highest term is null in the first case. We added an asterisk to the instance names for which both ACI and ALNS succeeded in scheduling all the interventions. For these instances, the last row then shows the average values of *Mi2Av*, *Mi2UB* and *Mi2Mi*, and the average value of processing time of ALNS.

For the eight instances in which both ACI and ALNS succeeded in scheduling all the interventions, the ALNS outperforms the ACI approach. The averaged *Mi2Av* is about halved, the *Mi2UB* is improved by about 10.2%, and the *Mi2Mi* is improved by about 11.2%. The processing times are shorter than the timeout fixed in Riise et al. (2016). For the two instances for which the ACI approach failed to schedule all the interventions the ALNS succeeded. For the a_08 instance, it takes about five minutes and, for the a_05 instance, it takes about 20 minutes. Better results are obtained for each instance, all the patients' interventions are scheduled which had not been done using the ACI algorithm.

## 7. Conclusions and future work

In this paper, we address the patient admission problem for which surgical interventions of hundreds of patients have to be scheduled within a horizon based on several months. The issue faced by hospital planners is to schedule all

the interventions while considering operating room and surgeon availability, in addition to the constraints related to each patient's surgical intervention. We introduce a two-phase 2PSC-EM heuristic and an Adaptive Large Neighborhood Search (ALNS) approach, both dedicated to the patient admission problem. The ALNS makes use of the adaptive mechanisms for destruction, construction and acceptance procedures that we investigate for the patient admission problem. A good trade-off between an adaptive diversification mechanism and an adaptive acceptance mechanism is needed so as not to waste processing time by exploring unnecessary low quality solutions.

Computational experiments are conducted on benchmark instances from the literature. Computational results show that the 2PSC-EM heuristic achieved better results than a general purpose approach not dedicated to the admission problem, whereas both heuristics fail in scheduling all interventions for some instances. The experiments concerning the parameter tuning of the ALNS approach can be easily conducted to obtain a good trade-off between solution quality and processing time. Computational experiments are to evaluate the effectiveness of each component of the ALNS approach, and all the components are necessary to obtain good solutions within good processing times.

The ALNS approach obtains better results for each instance, i.e, all the interventions are scheduled and the patients' health constraints for interventions are better respected.

One instance required about 20 minutes, all other instances are processed in a hundred seconds. It would be interesting to investigate new solution methods to reduce processing time. One possible research trail is to experiment other approaches based on other metaheuristics or by integrating learning methods as overviewed in Karimi-Mamaghan et al. (2022), Bengio et al. (2021) and Queiroz dos Santos et al. (2014).

A future research direction would be to consider additional objectives and constraints on minimum and/or maximum workloads for surgeons or on operating rooms use to obtain balanced schedules with regard to the use of resources. Fairness between surgeons is a sensitive issue for a hospital organization and load balancing of the rooms may also be useful.

**Acknowledgements**

This work was supported by the OIILH project (Optimisation Inter et Intra Logistique Hospitalière) managed by the National Agency for Research (Reference ANR-18-CE19-0019). It was also supported by the Labex MS2T, funded by the French Government, via the program "Investments for the future" managed by the National Agency for Research (Reference ANR-11-IDEX-0004-02).

**Disclosure statement**

The authors report there are no competing interests to declare.

**Data availability statement**

The data that support the findings of this study are openly available in "Generalised Surgery Scheduling Data" at http://www.sintef.no/projectweb/healthcare-optimization/testbed.


Akbarzadeh, B., Moslehi, G., Reisi-Nafchi, M., Maenhout, B., 2020. A diving heuristic for planning and scheduling surgical cases in the operating room department with nurse re-rostering. Journal of Scheduling 2020 23:2 23, 265–288. doi:10.1007/S10951-020-00639-6.

Almeida, B.F., Correia, I., Saldanha-Da-Gama, F., 2016. Priority-based heuristics for the multi-skill resource constrained project scheduling problem. Expert Systems with Applications 57, 91–103.

Amarouche, Y., Guibadj, R.N., Chaalal, E., Moukrim, A., 2020. Effective neighborhood search with optimal splitting and adaptive memory for the team orienteering problem with time windows. Computers and Operations Research 123, 105039. doi:10.1016/j.cor.2020.105039.

Ballestín, F., Pérez, A., Quintanilla, S., 2019. Scheduling and rescheduling elective patients in operating rooms to minimise the percentage of tardy patients. Journal of Scheduling 22, 107–118. doi:10.1007/S10951-018-0570-4.

Ben-Said, A., El-Hajj, R., Moukrim, A., 2019. A variable space search heuristic for the capacitated team orienteering problem. Journal of Heuristics 25, 273–303.

Bengio, Y., Lodi, A., Prouvost, A., 2021. Machine learning for combinatorial optimization: A methodological tour d'horizon. European Journal of Operational Research 290, 405–421. doi:10.1016/j.ejor.2020.07.06.

Cardoen, B., Demeulemeester, E., Beliën, J., 2010. Operating room planning and scheduling: A literature review. European Journal of Operational Research 201, 921–932.

Castro, P.M., Marques, I., 2015. Operating room scheduling with generalized disjunctive programming. Computers & Operations Research 64, 262 – 273.

Denton, B., Viapiano, J., Vogl, A., 2007. Optimization of surgery sequencing and scheduling decisions under uncertainty. Health Care Management Science 10, 13–24.

Dueck, G., 1993. New optimization heuristics; The great deluge algorithm and the record-to-record travel. Journal of Computational Physics 104, 86–92.

Erdogan, S.A., Denton, B.T., 2011. Surgery Planning and Scheduling. American Cancer Society. chapter 1. pp. 1–10.

Fischer, G.S., da Rosa Righi, R., de Oliveira Ramos, G., da Costa, C.A., Rodrigues, J.J., 2020. Elhealth: Using internet of things and data prediction for elastic management of human resources in smart hospitals. Engineering Applications of Artificial Intelligence 87, 103285. doi:https://doi.org/10.1016/j.engappai.2019.103285.

Gupta, D., 2009. Surgical suites' operations management. Production and Operations Management 16, 689–700.

Hashemi Doulabi, S.H., Rousseau, L.M., Pesant, G., 2016. A constraint-programming-based branch-and-price-and-cut approach for operating room planning and scheduling. INFORMS Journal on Computing 28, 432–448.

Hemmati, A., Hvattum, L.M., 2017. Evaluating the importance of randomization in adaptive large neighborhood search. International Transactions in Operational Research 24, 929–942.

Hussain, K., Mohd Salleh, M.N., Cheng, S., Shi, Y., 2019. Metaheuristic research: a comprehensive survey. Artificial intelligence review 52, 2191–2233.

Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A.M., Talbi, E.G., 2022. Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. European Journal of Operational Research 296, 393–422. doi:https://doi.org/10.1016/j.ejor.2021.04.032.

Kolisch, R., Hartmann, S., 1999. Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis, in: Project scheduling. Springer, pp. 147–178.

Lan, S., Fan, W., Yang, S., Pardalos, P.M., Mladenovic, N., 2021. A survey on the applications of variable neighborhood search algorithm in healthcare management. Annals of Mathematics and Artificial Intelligence 89, 741–775.

Magerlein, J.M., Martin, J.B., 1978. Surgical demand scheduling: a review. Health services research 13 4, 418–33.

Mara, S.T.W., Norcahyo, R., Jodiawan, P., Lusiantoro, L., Rifai, A.P., 2022. A survey of adaptive large neighborhood search algorithms and applications. Computers & Operations Research , 105903.

May, J.H., Spangler, W.E., Strum, D.P., Vargas, L.G., 2011. The surgical scheduling problem: Current research and future opportunities. Production and Operations Management 20, 392–405. doi:https://doi.org/10.1111/j.1937-5956.2011.01221.x.

Oliveira, M., Bélanger, V., Marques, I., Ruiz, A., 2020. Assessing the impact of patient prioritization on operating room schedules. Operations Research for Health Care 24, 100232.

Park, J., Kim, B.I., Eom, M., Choi, B.K., 2021. Operating room scheduling considering surgeons' preferences and cooperative operations. Computers & Industrial Engineering 157, 107306.

Queiroz dos Santos, J.P., de Melo, J.D., Duarte Neto, A.D., Aloise, D., 2014. Reactive search strategies using reinforcement learning, local search algorithms and variable neighborhood search. Expert Systems with Applications 41, 4939–4949. doi:https://doi.org/10.1016/j.eswa.2014.01.040.

Rahimi, I., Gandomi, A.H., 2020. A comprehensive review and analysis of operating room and surgery scheduling. Archives of Computational Methods in Engineering , 1–22.

Riise, A., Mannino, C., 2012. The surgery scheduling problem–a general model. https://www.sintef.no/en/publications/publication/1268319/. Report.

Riise, A., Mannino, C., Burke, E.K., 2016. Modelling and solving generalised operational surgery scheduling problems. Computers & Operations Research 66, 1–11.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40, 455–472.

Roshanaei, V., Naderi, B., 2021. Solving integrated operating room planning and scheduling: Logic-based benders decomposition versus branch-price-and-cut. European Journal of Operational Research 293, 65–78.

Samudra, M., Riet, C.V., Demeulemeester, E., Cardoen, B., Vansteenkiste, N., Rademakers, F.E., 2016. Scheduling operating rooms: achievements, challenges and pitfalls. Journal of Scheduling 19, 493–525.

Sier, D., Tobin, P., McGurk, C., 1997. Scheduling surgical procedures. Journal of the Operational Research Society 48, 884–891. doi:10.1057/palgrave.jors.2600441.

SINTEF, 2013. Testbed instances for optimization in health care. http://www.sintef.no/projectweb/health-care-optimization/testbed.

Tormos, P., Lova, A., 2003. An efficient multi-pass heuristic for project scheduling with constrained resources. International Journal of Production Research 41, 1071–1086. doi:10.1080/0020754021000033904.

Wang, T., Meskens, N., Duvivier, D., 2015. Scheduling operating theatres: Mixed integer programming vs. constraint programming. European Journal of Operational Research 247, 401 – 413.

Xiang, W., 2017. A multi-objective ACO for operating room scheduling optimization. Natural Computing 16, 607–617.

Xiang, W., Yin, J., Lim, G., 2015. An ant colony optimization approach for solving an operating room surgery scheduling problem. Computers & Industrial Engineering 85, 335 – 345.

Xie, X., Lawley, M.A., 2015. Operations research in healthcare. International Journal of Production Research 53, 7173–7176.

Zhang, J., Dridi, M., El Moudni, A., 2019. A two-level optimization model for elective surgery scheduling with downstream capacity constraints. European Journal of Operational Research 276, 602–613. doi:https://doi.org/10.1016/j.ejor.2019.01.036.

Zhou, Y., Parlar, M., Verter, V., Fraser, S., 2021. Surgical Scheduling with Constrained Patient Waiting Times. Production and Operations Management 30, 3253–3271. doi:10.1111/poms.13427.