A STOCHASTIC PROGRAMMING APPROACH TO SURGERY SCHEDULING UNDER PARALLEL PROCESSING PRINCIPLE

Batuhan Celik[1], Serhat Gul[2], Melih Celik[3]

[1]Department of Industrial Engineering, Bilkent University, Ankara, Turkey

[2]Department of Industrial Engineering, TED University, Ankara, Turkey

[3]School of Management, University of Bath, Bath, UK

Correspondence

Serhat Gul, Department of Industrial Engineering

TED University, 06420, Çankaya, Ankara, Turkey

E-mail: serhat.gul@tedu.edu.tr

A Stochastic Programming Approach to Surgery Scheduling under Parallel Processing Principle

Abstract

Parallel processing is a principle which enables simultaneous implementation of anesthesia induction and operating room (OR) turnover with the aim of improving OR utilization. In this article, we study the problem of scheduling surgeries for multiple ORs and induction rooms (IR) that function based on the parallel processing principle under uncertainty. We propose a two-stage stochastic mixed-integer programming model considering the uncertainty in induction, surgery and turnover durations. We sequence patients and set appointment times for surgeries in the first stage and assign patients to IRs at the second stage of the model. We show that an optimal myopic policy can be used for IR assignment decisions due to the special structure of the model. We minimize the expected total cost of patient waiting time, OR idle time and IR idle time in the objective function. We enhance the model formulation using bounds on variables and symmetry-breaking constraints. We implement a novel progressive hedging algorithm by proposing a penalty update method and a variable fixing mechanism. Based on real data of a large academic hospital, we compare our solution approach with several scheduling heuristics from the literature. We assess the additional benefits and costs associated with the implementation of parallel processing using near-optimal schedules. We examine how the benefits are inflated by increasing the number of IRs. Finally, we estimate the value of stochastic solution to underline the importance of considering uncertainty in durations.

Keywords: surgery scheduling, stochastic programming, parallel processing, progressive hedging

Funding: This material is based in part on work supported by the TED University Institutional Research Fund [Grant T-18-B2010-33020].

1 Introduction

Operating room (OR) expenses account for a significant portion of total costs in a hospital (Gul et al. 2015). *Parallel surgery processing* has recently been implemented in some hospitals to improve OR utilization, and hence reduce costs (Marjamaa et al. 2009). Parallel processing is a technique that allows concurrent implementation of *anesthesia induction* and *turnover*. This implies that the surgical process for a patient can start by giving an anesthetic agent to the patient in the preoperative holding area while the OR is being cleaned and set up for the patient. In other words, it is not mandatory to finish the whole process related to the preceding surgery before giving anesthetic agent to the current patient. Contrary to conventional practice (i.e., *serial processing*), OR is used only for *incision* and closing the incision (i.e., *postincision*) in parallel processing. Therefore, anesthesia induction (i.e., *preincision*) is handled in a room usually called as *induction room* (IR) in the preoperative holding area.

The benefit of parallel processing has been well acknowledged by the medical community thanks to the actual experiments conducted in several different hospitals. The experiments show that the surgical cases in an OR finish earlier in the day if parallel processing is preferred over serial processing (Hanss et al. 2005, Torkki et al. 2005, Harders et al. 2006, Friedman et al. 2006, Smith et al. 2006). Therefore, the number of surgeries that can be performed in a day increases under this setting. For example, the time without operation in the ORs is decreased by 45.6% and 54% in the experiments by Torkki et al. (2005) and Harders et al. (2006), respectively. El-Boghdadly et al. (2020) review 15 studies that involve approximately 8900 patients in the trials in total, and indicate that parallel processing is favored in terms of clinical outcomes. Furthermore, they report that the OR throughput per day increases by 1.7 cases when parallel processing is implemented. Torkki et al. (2005) mention that the benefit of parallel processing would be augmented if surgery schedules are systematically created. However, the optimization models and solution methods that have been developed to solve surgery scheduling problems in the literature have not considered the challenges associated with parallel processing.

Designing surgery schedules is difficult due to uncertainty in surgery durations. In serial processing, surgery duration refers to the summation of preincision, incision and postincision durations (Batun et al. 2011). However, since induction is conducted in a separate location in parallel processing, surgery duration represents the summation of only incision and postincision durations in this study. Uncertainty in induction and OR turnover (i.e., cleaning and set up) times creates

additional challenges for the scheduler. Furthermore, the decisions affect the trade-off between patient and provider-related performance measures such as patient waiting time, IR idle time and OR idle time.

In this article, we study the problem of scheduling surgeries under uncertainty for a surgical suite consisting of multiple ORs and IRs that work based on parallel processing principle. The uncertainty exists due to induction, surgery and turnover durations. We model our problem as a two-stage stochastic mixed integer programming (SMIP) formulation. The main decisions in the model include sequencing patients in the daily appointment list and setting appointment times. We also consider the limited availability of induction rooms. The assignment of patients to induction rooms are the decisions made at the second stage for each realized scenario. We minimize the expected total cost of patient waiting time, OR idle time, and IR idle time in the objective function. We propose a progressive hedging algorithm (PHA) that benefits from the underlying problem structure to design near-optimal schedules. We conduct computational experiments to illustrate the performance of the PHA solutions in comparison to the optimal solutions. The experiments are conducted based on real data of a major hospital in the United States. We compare the proposed method with well-known scheduling heuristics that are used to create daily surgery schedules. We compare parallel processing with serial processing based on near-optimal schedules in terms of patient waiting time and OR closure times. We assess the role of the number of IRs while implementing parallel processing. Finally, we estimate the value of stochastic solution to show the benefit of considering uncertainty in durations while scheduling surgeries under parallel processing principle.

The organization of the following sections of the article is as follows. In Section 2, the relevant articles are reviewed, and the principal contributions of our study are provided. In Section 3, the problem is described and model formulation is extensively discussed. In Section 4, the proposed PHA algorithm is presented in detail. The results of computational experiments are shown in Section 5. Finally, conclusions and possible extensions are provided in Section 6.

2 Literature Review

We review three categories of articles to distinguish our study from the other studies in the literature: studies (i) considering appointment scheduling decisions for a daily list of patients, (ii) focusing on benefits obtained through concurrent implementation of anesthesia induction and turnover

by parallel processing principle, and (iii) defining parallel processing as the setting where a single surgeon works in two different ORs in an alternating fashion.

Among the first category of studies, Gupta and Denton (2008) conduct a literature review on appointment time scheduling problems and group the studies based on the settings considered. These settings are primary care, specialty care, and elective surgery care. We only present studies that consider elective surgery care. Studies of the remaining two groups are extensively discussed in Cayirli and Veral (2003), Gupta and Denton (2008) and Ahmadi-Javid et al. (2017). Gul et al. (2011) use heuristics to sequence surgeries and set patient appointment times with the objective of minimizing expected patient waiting time and surgical suite overtime. For setting patient appointment times, they use a job hedging procedure, where the allocated duration for a surgery is calculated based on a certain percentile of the surgery duration distribution.

Denton and Gupta (2003) formulate the appointment scheduling problem for a single-server case as a two-stage stochastic linear program with the objective of minimizing the expected total cost of customer waiting time, server idle time, and tardiness. Their model is generalized to represent any service system where appointment scheduling would be relevant. However, their study is motivated by a surgery scheduling problem in a single OR. In the context of surgery scheduling, the performance measures correspond to patient waiting time, OR idle time, and OR overtime, respectively. They solve the resulting formulation using an L-shaped algorithm with sequential bounding. Khaniyev et al. (2020) focus on the same problem and propose simple heuristics that are shown to perform well for a daily sequence of surgeries with non-identical duration distributions.

Denton et al. (2007) extend the model in Denton and Gupta (2003) by also considering sequencing decisions. They formulate the problem as a two-stage stochastic mixed integer program. They make sequencing decisions based on heuristics and solve the reduced version of the problem to set patient appointment times. Mancilla and Storer (2012) provide a reformulation of the model in Denton et al. (2007), and implement a heuristic based on Benders' decomposition to solve the model.

Zhang and Xie (2015) study how to set appointment times for surgeries in a multiple-OR setting. The objective is to minimize the total expected cost of surgeon waiting, OR idle, and OR overtime. They formulate a simulation-based optimization model. They propose stochastic gradient algorithm to solve the model.Vandenberghe et al. (2019) study a stochastic surgery sequencing problem also for a multiple-OR case by assuming that the OR assignment decisions are already made. They call their problem as the stochastic break-in moment problem since they minimize the expected

maximum waiting time of an arriving emergent surgery. They solve the model using sample average approximation algorithm and different heuristics. Atighehchian et al. (2020) first allocate surgeries to ORs, surgery assistants and residents by considering the setting of teaching hospitals. They then sequence surgeries for each OR and resident without setting any appointment times. The objective of the model minimizes expected OR idle time and overtime costs. They solve the resulting two-stage stochastic programming model using an L-shaped method.

Lee and Yih (2014) set patient appointment times for multiple ORs by considering the availability of post-anesthesia care unit (PACU) resources and assuming a fixed patient sequence. The problem is formulated as flexible job shop model with fuzzy sets which are used to model the uncertainty in process durations. The criteria include patient waiting time, OR idle time, and surgical suite completion time in the study. They use a combination of heuristic and genetic algorithms to find solutions. Bai et al. (2017) also model the appointment scheduling problem for multiple ORs considering PACU constraints. Their model also does not allow changes in the order of surgeries. They formulate the problem as a discrete-event dynamic system-based stochastic optimization model. They minimize the expected total cost of patient waiting time, OR idle time, OR blocking time, OR overtime and PACU overtime. To solve the deterministic equivalent of their model, they propose a gradient based algorithm. Varmazyar et al. (2020) study patient sequencing problem for a setting also including multiple ORs and PACU resources. They estimate surgery and PACU durations using continuous phase-time distributions and propose a heuristic to solve the problem.

Erdogan and Denton (2013) formulate a dynamic appointment scheduling problem using a multi-stage stochastic linear program. They schedule appointment requests dynamically. Their study is motivated by the arrival of urgent add-on cases that require revising the current schedule dynamically. They solve the model using decomposition-based algorithms.

Our study focuses on appointment scheduling decisions as the first category of articles reviewed. However, we consider the flow of patients through a surgical suite including multiple IRs and ORs that operate based on parallel processing principle.

In the second category of our literature review, we only identify the study of Marjamaa et al. (2009). They investigate four different parallel processing systems in which the induction and turnover are carried out simultaneously using operations research methods. They compare their models with serial processing models using discrete-event simulation. All models are assumed to have four ORs. In the first model, there is an IR adjacent to each OR. In this model, there is

an additional induction team consisting of two nurses for each OR. While a team is operating in the OR, the other team with an anesthesiologist deals with the patient in the IR. In the second model, the team consisting of an anesthesiologist and two nurses induce anesthesia to all patients in different IRs. In the third model, there is a central IR, where there are three beds and an induction team. In the last model, there are three ORs used for surgeries and one OR used for induction. The results are provided in terms of the total number of completed surgeries, over-utilization and under-utilization of ORs, and personnel expenses. All parallel processing systems yield lower costs than serial processing systems. The study also investigates the possible benefit of assigning all short surgeries to one OR and the long surgeries to another OR.

Marjamaa et al. (2009) investigate parallel processing systems only using simulation. On the other hand, we develop a two-stage SMIP model to find near-optimal schedules for surgical suites that operate based on parallel processing.

The third category of studies investigate a different form of parallel processing where a single surgeon performs parallel surgeries in two ORs. Once the incision process of the surgery is completed, the surgeon moves to the other OR and starts dealing with the incision phase of that patient. However, the anesthesia inductions are applied in the ORs. Batun et al. (2011) and Mancilla and Storer (2013) are the two studies that propose such parallel processing implementations.

Batun et al. (2011) decide on the number of ORs that would be used during the day, assign surgeries to ORs, sequence surgeries, and set the start time of the first surgery of each surgeon in a day. It is assumed that the surgeons operate their surgeries in series. Hence, the appointment time for a particular surgery is not needed to be set. The problem is modeled as an SMIP and solved by an L-shaped algorithm. Mancilla and Storer (2013) study the assignment of surgeries to ORs and sequence surgeries. The problem is modeled as an SMIP and solved by a decomposition based method.

In our study, parallel processing represents a setting which is completely different from the one considered in Batun et al. (2011) and Mancilla and Storer (2013). The differences of our study can be summarized as follows: An induction cannot be applied in an OR. An IR, located adjacent to an OR in the preoperative holding area, is used for this task instead. The operating discipline is considered to be parallel because the induction for a patient in an IR and the turnover after the surgery of the preceding patient in an OR are performed simultaneously. A surgeon works only in a single OR and does not have to alternate between ORs for the implementation of parallel processing. Furthermore, surgeons do not have to operate their own surgeries consecutively. Therefore, we set

appointment times for each surgery in our study.

3 Problem Description and Model Formulation

We study the problem of sequencing patients and setting appointment times for a daily surgery list of a surgical suite having multiple ORs and IRs that function according to the parallel processing principle. Therefore, the induction is not applied in an OR, but in an IR so that the patient can be ready for the incision part of the surgery while the OR is cleaned and set up after the previous surgery. We assume that patients are punctual and arrive to the surgical suite exactly at their appointment times. They first wait for an IR to become available if all are busy. Then, they can be assigned to any one of the available IRs, as the IRs are assumed to be identical. After the induction is applied by an anesthesiologist in the IR, the patient is taken to an OR for surgery, possibly after some waiting. We ignore the transfer time from an IR to an OR. No OR assignment decision is made in the problem since we assume that patient-to-OR assignments are known a priori. After the surgery is performed in the OR, the turnover starts to make the OR ready for the next patient.

We assume that an anesthesiologist provides care only for a set of patients whose surgeries are planned to be performed in the same OR. Since the OR assignments are already known, this means the anesthesiologist of each patient is also known. Furthermore, we assume the anesthesiologist can start the induction for a patient in an IR even if the surgery of the preceding patient continues in the OR that the anesthesiologist is responsible for. This is because the anesthesiologist does not need to be active during the whole surgery process of a patient as their aides can also monitor the patient in the OR. On the other hand, we assume that an anesthesiologist must be present in the IR over the whole induction process for a patient.

We formulate the problem as a two-stage SMIP. At the first stage of the model, patients are sequenced, and their appointment times are set under uncertainty in induction, surgery and turnover durations. At the second stage, patients are assigned to one of the identical IRs based on a *myopic policy* that is optimal due to the special structure of the two-stage SMIP (as will be explained after the model is provided). The myopic policy allows all IR assignment decisions to be made at a single stage. The performance measures including IR and OR idle times, and patient waiting times for IR and OR are also calculated in the second stage. Besides, some auxiliary variables including surgery start times, IR closure times and OR closure times are also calculated at the same stage.

The waiting time of a patient for an IR is equal to the difference between the induction start

time and appointment time of the patient. The induction start of a patient may be delayed because the induction of preceding patient may not finish on time. In that case, the induction of the patient can start only when the preceding patient is taken to the OR for surgery. This implies that the patients stay in an IR after induction when they need to wait for an OR to become available. The waiting time of a patient for an OR is equal to the difference between the surgery start time and induction finish time of the patient. The surgery of a patient may start late because the turnover for the preceding surgery may finish later than planned.

Idle time of an IR is equal to the difference between the IR closure time and summation of induction durations of all patients treated in that IR. Note that an IR is assumed to be idle even if a patient occupies the room as he/she waits for an OR to be available, because the IR staff is kept idle during that period. This implies that such idleness cases are punished twice due to their negative impact into both provider and patient related performance measures. An IR closes when the last patient treated in the IR is taken from the IR to an OR for surgery. Therefore, IR closure time is equal to the surgery start time of the last patient treated in the IR. Idle time of an OR is equal to the difference between the OR closure time and summation of surgery and turnover durations of all patients treated in the OR. An OR closes when the turnover for the last patient treated in the OR is finished.

To illustrate the flow of operations in the surgical suite and explain the performance measures clearly, Figure 1 shows an example 4-hour schedule with seven patients, two IRs, and three ORs. Patients 1 and 2 are pre-assigned to OR 1, patients 3 and 4 to OR 2, and the remaining patients to OR 3. The appointment times are $a_1 = 20$, $a_2 = a_7 = 0$, $a_3 = 55$, $a_4 = 10$, $a_5 = 38$, and $a_6 = 87$. In the beginning of the shift, patients 2 and 7 are assigned to IRs 2 and 1, respectively. Patient 4 arrives during the induction of these two patients, and has to wait for 7 minutes until patient 7 finishes induction and moves to OR 3. Similarly, patient 1 arrives while the induction of patients 2 and 4 are in progress, and needs to wait for 14 minutes before the induction of patient 2 ends and surgery starts in OR 1. One minute later, the induction of patient 4 finishes and the patient moves to OR 2. Patients 5, 3 and 6 arrive later in order, and the only other waiting for the IR is patient 3, for 1 minute.

Figure 1 also shows examples of parallel processing in the suite. For example, the induction of patient 5 is underway while the turnover in its assigned OR is in progress after the surgery of patient 7 is complete. A similar observation can be made for patient 3 in IR 2, at the same time the turnover in OR 2 is ongoing prior to the surgery of this patient.

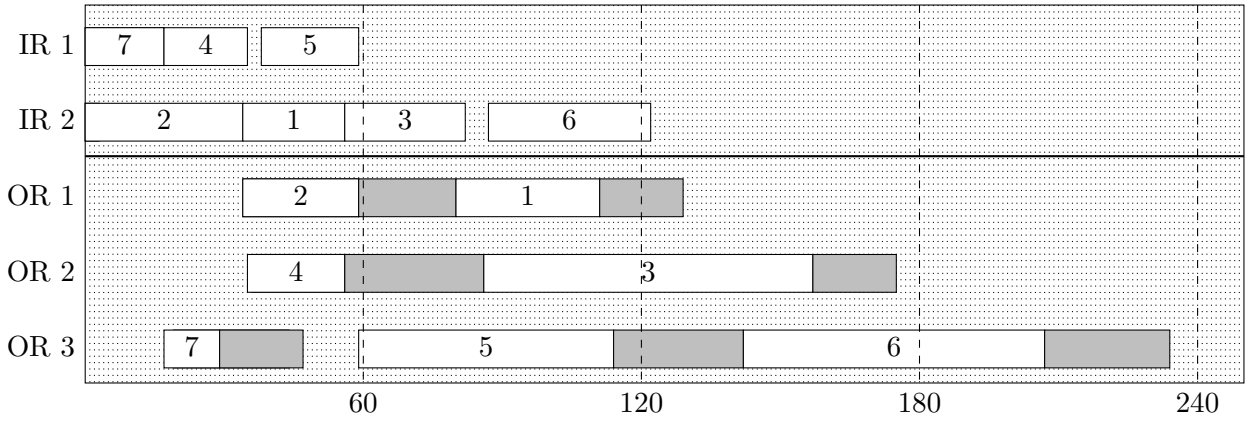

Figure 1: An example 4-hour IR and OR schedule, where each number corresponds to a patient, white bars to the duration of induction and surgeries, and dark bars to turnover durations

For the example in Figure 1, the total waiting time for the IR is 22 minutes (14, 1, and 7 minutes for patients 1, 3, and 4, respectively). Patients need to wait for a total of 48 minutes for the ORs (24, 4, and 20 minutes for patients 1, 3, and 6, respectively). This yields a total waiting time of 70 minutes. IR 1 closes after 59 minutes, when patient 1 moves to OR 1. IR 2 closes after 142 minutes, after OR 3 becomes available for patient 6. Similarly, ORs 1, 2 and 3 close after 142, 175 and 234 minutes, respectively, following the turnover of their last patients. Total OR idle time is 98 minutes, with 34, 35 and 29 minutes of idle time at OR 1, OR 2, and OR 3, respectively. IR 1 has a total idle time of 4 minutes between the departure of patient 4 for OR 2 and the arrival of patient 5. IR 2 has a total idle time of 25 minutes; 5 minutes between the departure of patient 3 for OR 2 and the arrival of patient 6, and 20 minutes when patient 6 needs to wait for the surgery of patient 5 to end in OR 3.

The notation we use for the formulation of the problem is provided in Table 1. Using the sets, parameters and decision variables in the table, we formulate the problem using the following two-stage SMIP:

$$\min \quad \mathcal{Q}(\mathbf{a},\mathbf{u}) \tag{1}$$

$$s.t. \quad a_j \geq a_i - M(1-u_{ij}) \quad \forall i,j \in S, j \neq i \tag{2}$$

$$u_{ij} + u_{ji} = 1 \quad \forall i,j \in S, j > i \tag{3}$$

$$a_i \in \mathbb{Z}^+ \quad \forall i \in S \tag{4}$$

$$u_{ij} \in \{0,1\} \quad \forall i,j \in S, j \neq i \tag{5}$$

Table 1: Notation used throughout the mathematical model

Index sets	
S	Surgeries (patients)
K	Induction rooms
R	Operating rooms
S_r	Surgeries to be conducted in OR $r \in R$
Ω	Scenarios
Deterministic Parameters	
c^I	Cost of idle OR time per minute
c^J	Cost of idle IR time per minute
c^W	Cost of patient waiting time per minute
M	A large value
Stochastic Parameters	
$d_i(\omega)$	Duration of surgery for patient $i \in S$ under scenario $\omega \in \Omega$
$e_i(\omega)$	Duration of induction for patient $i \in S$ under scenario $\omega \in \Omega$
$q_i(\omega)$	Duration of turnover after the surgery for patient $i \in S$ under scenario $\omega \in \Omega$
First-Stage Decision Variables	
a_i	Appointment time of patient $i \in S$
u_{ij}	$= \begin{cases} 1, & \text{if patient } i \in S \text{ precedes patient } j \in S \text{ (general precedence)} \\ 0, & \text{otherwise} \end{cases}$
Second-Stage Decision Variables	
$y_{ik}(\omega)$	$= \begin{cases} 1, & \text{if patient } i \in S \text{ is assigned to IR } k \in K \text{ under scenario } \omega \in \Omega \\ 0, & \text{otherwise} \end{cases}$
$A_i(\omega)$	Surgery start time for patient $i \in S$ under scenario $\omega \in \Omega$
$W_i(\omega)$	Waiting time for patient $i \in S$ for an OR under scenario $\omega \in \Omega$
$Y_i(\omega)$	Waiting time for patient $i \in S$ for an IR under scenario $\omega \in \Omega$
$I_r(\omega)$	Idle time of OR $r \in R$ under scenario $\omega \in \Omega$
$J(\omega)$	Total idle time of all IRs under scenario $\omega \in \Omega$
$F_r(\omega)$	Closure time of OR $r \in R$ under scenario $\omega \in \Omega$
$G_k(\omega)$	Closure time of IR $k \in K$ under scenario $\omega \in \Omega$

where $\mathcal{Q}(\mathbf{a},\mathbf{u}) = E_\xi[Q(\mathbf{a},\mathbf{u},\xi(\omega))$ is the expected second-stage cost, and $Q(\mathbf{a},\mathbf{u},\xi(\omega)))$ is determined by:

$$\min \quad c^J J(\omega) + c^I \sum_{r\in R} I_r(\omega) + c^W \sum_{i\in S} \big(W_i(\omega) + Y_i(\omega)\big) \tag{6}$$

$$s.t. \quad \sum_{k\in K} y_{ik}(\omega) = 1 \qquad \forall i \in S \tag{7}$$

$$a_j + Y_j(\omega) + M(1-u_{ij}) \geq a_i + Y_i(\omega) \qquad \forall i,j \in S, j \neq i \tag{8}$$

$$a_j + Y_j(\omega) + M(1-u_{ij}) + M(2 - y_{ik}(\omega) - y_{jk}(\omega)) \geq A_i(\omega) \qquad \forall i,j \in S, j \neq i, \forall k \in K \tag{9}$$

$$a_j + Y_j(\omega) + M(1-u_{ij}) \geq a_i + Y_i(\omega) + e_i(\omega) \qquad \forall i,j \in S_r, j \neq i, \forall r \in R \tag{10}$$

$$A_i(\omega) = a_i + Y_i(\omega) + e_i(\omega) + W_i(\omega) \qquad \forall i \in S \tag{11}$$

$$A_j(\omega) + M(1-u_{ij}) \geq A_i(\omega) + d_i(\omega) + q_i(\omega) \qquad \forall i,j \in S_r, j \neq i, \forall r \in R \tag{12}$$

$$F_r(\omega) \geq A_i(\omega) + d_i(\omega) + q_i(\omega) \qquad \forall i \in S_r, \forall r \in R \tag{13}$$

$$I_r(\omega) = F_r(\omega) - \sum_{i\in S_r} (d_i(\omega) + q_i(\omega)) \qquad \forall r \in R \tag{14}$$

$$G_k(\omega) \geq A_i(\omega) - M(1 - y_{ik}(\omega)) \qquad \forall i \in S, \forall k \in K \tag{15}$$

$$J(\omega) = \sum_{k\in K} G_k(\omega) - \sum_{i\in S} e_i(\omega) \tag{16}$$

$$A_i(\omega), W_i(\omega), Y_i(\omega) \geq 0 \qquad \forall i \in S \tag{17}$$

$$I_r(\omega), F_r(\omega), G_k(\omega), J(\omega) \geq 0 \qquad \forall r \in R, \forall k \in K \tag{18}$$

$$y_{ik}(\omega) \in \{0,1\} \qquad \forall i \in S, \forall k \in K \tag{19}$$

Assuming that the uncertainty in durations can be represented by a finite set of scenarios, the objective function (1) only minimizes the expected second-stage cost over all scenarios, which is equivalent to the total expected cost of patient waiting for IR and OR, IR idle time, and OR idle time.

The first-stage constraints are shown by the constraint set (2)-(5). Constraints (2) and (3) define the sequencing rules for patients. Constraint (2) ensures that if patient $i \in S$ precedes patient $j \in S$ in the daily patient list, then the appointment time of patient i cannot be set later than that of patient j. Constraint (3) provides that either patient $i \in S$ precedes patient $j \in S$ in the daily patient list or vice versa. Constraints (4) and (5) represent integrality and binary restrictions on the first-stage variables, respectively. We represent a_i as integer variables rather

than continuous variables due to two reasons. First, as will be evident when the variable fixing procedure is explained in Section 4.3.2, the PHA converges much faster in the case that integer first-stage variables are used. Second, we aim to generate realistic and implementable schedules by proposing integer appointment times (corresponding to minutes) as in practice.

The second-stage objective function for given first-stage decision variables and a particular scenario is formulated explicitly by (6). It minimizes the total cost of patient waiting for IR and OR, IR idle time, and OR idle time for a given scenario.

Constraint (7) ensures that each patient is assigned to exactly one IR. Constraint (8) ensures that the induction start time of patient $j \in S$ is greater than or equal to that of patient $i \in S$ only in case patient j comes after patient i in the patient sequence. Constraint (9) applies only to the patients that are assigned to the same IR. The constraint guarantees that the induction for patient $j \in S$ can start only after patient $i \in S$ is moved from the IR and placed to her OR in case patient j comes after patient i in the patient sequence.

Constraint (10) enforces a restriction for the patients who are pre-assigned to the same OR. Due to one-to-one matching between ORs and anesthesiologists, the constraint is valid for the patients who are pre-assigned to the same anesthesiologist. For a given set of surgeries to be treated in OR $r \in R$, it ensures that the induction for patient $j \in S_r$ can start after that for patient $i \in S_r$ ends, if patient i precedes patient j in the patient sequence. The constraint implies that each anesthesiologist can perform a single induction at a time even if the patients are treated in separate IRs.

Constraint (11) is formulated to determine the amount of patient waiting time for an OR right after the induction ends. The constraint sets the surgery start time for patient $i \in S$ equal to the summation of induction completion time of patient i and waiting time of patient i for an OR. Constraint (12) ensures that surgery for a patient can start only if the turnover of the earlier surgeries in the same OR ends. The constraint also implies that the patient sequence in the first stage is preserved at the second stage for each OR. Constraint (13) sets the actual closure time of an OR. It ensures that the closure time of OR $r \in R$ is not earlier than the turnover finish time of any surgery performed in OR r. Constraint (14) calculates the idle time in an OR. OR idle time is set equal to the actual OR closure time minus the total surgery and turnover durations for the patients treated in the OR. Constraint (15) defines the closure time of an IR. The closure time of an IR cannot be earlier than the surgery start times of the patients who go through that IR, as the patients wait for their surgeries in the IRs. Constraint (16) calculates the total idle time over all

IRs. This is achieved by subtracting the summation of induction durations from the summation of IR closure times. Finally, constraints (17)-(19) define non-negativity and binary restrictions on the second-stage variables.

We consider IR assignment decisions in the second stage, because considering them in the first stage would result in inefficiencies in the surgical suite as the IRs are identical. Even though the IR assignment decisions are made dynamically in the surgical suites as patients arrive for their appointments, the special structure of our model allows us to avoid formulating the problem as a multi-stage SMIP. We show that after the sequence of patients is fixed in the first stage, the decisions in the second stage can be made based on a myopic policy. Since using a myopic policy prevents the need of considering uncertainty related to induction, surgery and turnover durations, our problem can be formulated as a two-stage SMIP. The justification of our modeling choice is based on the following proposition, whose proof is provided in Appendix A.

Proposition 1 *For a given sequence of patients obtained from the first stage, each arriving patient is assigned to the first available induction room in the optimal schedule.*

The proof of proposition is valid mainly due to constraints (8). These constraints ensure that the sequence of patients in the first stage does not change during the treatments of the patients both in the IR and OR phases in the second stage. Hence, the optimal assignment of the patients to IRs can be achieved by a non-idling policy. This implies that all assignment decisions can be made at a single stage without formulating a dynamic model.

Note that the assumption of unchanged patient sequence is also practically relevant. Since we consider only elective surgeries in our problem, the urgency levels of the surgeries are all the same. Furthermore, the ORs to be used for each surgery are predetermined. Therefore, switching the planned position of two patients waiting for their surgeries would not be realistic considering the possible reactions of the patients. Such a change would be perceived by a patient as an unfair treatment, and hence the satisfaction level of the patients would be significantly affected in a negative direction. Therefore, the managers instead must ensure that the appointment times are carefully set such that the smooth flow of patients through the surgical suite can be maintained without making a change in the original sequence.

We strengthen the two-stage SMIP formulation by adding a number of valid inequalities into the second-stage problem, and solve the resulting formulation in the computational experiments. In particular, we impose bounds on some variables and formulate symmetry-breaking constraints.

We first propose a bound on the variables representing waiting times of patients for an OR. Suppose that patients $i, j, k \in S_r$ are the first, second and third patients to be treated in OR $r \in R$. The worst-case scenario from the perspective of patient j and k occurs when the induction finish times of patient j and k are equal to the surgery start time of patient i. Since patient j must wait until the OR is made ready after patient i leaves, the maximum amount that patient j waits for an OR is equal to the summation of surgery and turnover durations for patient i. Moreover, the maximum amount that patient k waits for an OR is equal to the summation of surgery and turnover durations for patients i and j. By generalizing this relationship for the patients to be treated in the same OR, we formulate the following bound on the $W_j(\omega)$ variables:

$$W_j(\omega) \leq \sum_{i \in S_r, i \neq j} \big(d_i(\omega) + q_i(\omega)\big) u_{ij} \qquad \forall j \in S_r, \forall r \in R \tag{20}$$

Next, we enforce that the closure time of the IR with index $(k-1) \in K$ is greater than or equal to the closure time of the IR with index $k \in K, k > 1$ using the following constraint:

$$G_{k-1}(\omega) \geq G_k(\omega) \qquad \forall k \in K \setminus \{1\} \tag{21}$$

The two-stage SMIP formulation can be further strengthened by setting the M values in Constraints (2), (8)-(10), (12), and (15) as tight as possible. Of these, the M values for Constraints (2) and (8)-(10) are bounded by the maximum induction start time of any patient. Similarly, for Constraints (12) and (15), the corresponding bound is the maximum possible surgery start time for any patient.

When the probability distributions of the random variables for surgery, induction, and turnover times have finite support, a valid upper bound for all M values can be found by assuming a single IR and OR and finding the maximum possible closure time of the OR. To do so, we set the stochastic parameters $d_i(\omega)$, $e_i(\omega)$ and $q_i(\omega)$ at their maximum possible levels $d_i^{max} = \max_{\omega \in \Omega} \{d_i(\omega)\}$, $e_i^{max} = \max_{\omega \in \Omega} \{e_i(\omega)\}$ and $q_i^{max} = \max_{\omega \in \Omega} \{q_i(\omega)\}$ for all patients $i \in S$. We then list the patients starting with $i = \arg\max_{i \in S} \{e_i^{max}\}$ and with the remaining patients in any order. Let $i = \langle 1 \rangle$ and $\langle |S| \rangle$ correspond to the index of the first and last patient in this list, respectively. We simulate the system as follows: Patient $i = \langle 1 \rangle$ enters the IR at $t = 0$, whereas all subsequent patients $\langle k \rangle \in S$ enter the IR and start their induction once the previous patient in the list leaves the IR. Patient $\langle k \rangle \in S$ enters the OR either immediately (if the OR is available) or as soon as turnover of $\langle k-1 \rangle \in S$ is

finished. Let the closure time of this system be F^{max}. The M value for Constraints (12) and (15) can then be set as $F^{max} - \min_{u \in S} \{(d_u^{max} + q_u^{max})\}$. The corresponding M value for Constraints (2) and (8)-(10) is given by $F^{max} - \min_{u \in S} \{(d_u^{max} + q_u^{max})\} - \min_{i \in S} \{\min_{\omega \in \Omega} \{e_i(\omega)\}\}$.

4 Solution Methodology

The number of variables and constraints would be quite large for realistically sized instances of SMIP models, since there are as many second-stage subproblems as the number of scenario realizations. The resulting complexity necessitates the application of sophisticated algorithms even when a near-optimal solution of a model is sufficient (Birge and Louveaux 2011). Therefore, various decomposition algorithms have been proposed to solve SMIP models. The L-shaped algorithm of Van Slyke and Wets (1969) is not applicable to the models including integer variables, which is also the case in our study. Laporte and Louveaux (1993) propose a variant of this algorithm, called integer L-shaped algorithm, to solve SMIP models having only binary variables in the first stage. Value function and set convexification based approaches are also developed in the literature to deal with models having binary variables in both stages (Sen and Higle 2005, Yuan and Sen 2009). However, they cannot be utilized to solve our model, because our model includes both general and binary integer variables in the first stage. Having also mixed-binary variables in the second stage makes the solution process particularly difficult for our model.

A class of algorithms that would be appropriate for solving our model is scenario decomposition-based methods. The PHA, which was proposed by Rockafellar and Wets (1991), is a commonly used scenario decomposition algorithm (Hvattum and Løkketangen 2009, Gonçalves et al. 2012, Gul et al. 2015, Demir et al. 2021). The algorithm is basically a type of augmented Lagrangian relaxation approach. To apply scenario decomposition at each iteration, a reformulation of the model is required before the implementation of the algorithm. Next, the nonanticipativity constraints (i.e., the constraints that enforce the restrictions that the first-stage variable values cannot take different values under different scenarios) in the reformulated model are relaxed and carried to the objective function using a penalty and Lagrangian term. The augmented Lagrangian relaxation approach allows decomposing the model into scenario subproblems. After solving scenario subproblems independently and then aggregating the scenario solutions over multiple iterations, the PHA provides a nonanticipative and feasible solution for the original model.

The PHA yields optimal solutions for convex models (Rockafellar and Wets 1991). Since the

two-stage SMIP model is nonconvex due to binary variables at both stages of the formulation, the algorithm provides a heuristic solution to our problem. Note that the PHA is suggested as a good heuristic for SMIP models in several studies in the literature (Crainic et al. 2011, Watson and Woodruff 2011, Gul et al. 2015, Demir et al. 2021).

We first provide the reformulation of the two-stage SMIP model in the remaining parts of this section. Next, we discuss the basic PHA with its general steps. Finally, we present our own implementation of the PHA which improves the performance of the basic PHA by linearizing the objective function, and using a variable fixing method and a penalty update mechanism.

4.1 Problem reformulation

To implement the PHA, a scenario separable formulation is needed. Therefore, we first reformulate the two-stage SMIP by explicitly representing nonanticipativity constraints in the model. For this end, we additionally define two decision variables: $a_i(\omega)$ as the appointment time for patient $i \in S$ under scenario $\omega \in \Omega$ and $u_{ij}(\omega)$ as a binary variable denoting whether patient $i \in S$ precedes patient $j \in S$ or not under scenario $\omega \in \Omega$.

Denoting the probability of occurrence of each scenario $\omega \in \Omega$ by p^ω, the reformulation of the two-stage SMIP, which we refer to as R-SMIP, is obtained as follows:

$$\min \quad \sum_{\omega\in\Omega} p^\omega \Big(c^J J(\omega) + c^I \sum_{r\in R} I_r(\omega) + c^W \sum_{i\in S} \big(W_i(\omega) + Y_i(\omega)\big)\Big) \tag{22}$$

$$s.t. \quad a_j(\omega) \geq a_i(\omega) - M(1-u_{ij}(\omega)) \qquad \forall i,j \in S, j \neq i, \forall \omega \in \Omega \tag{23}$$

$$u_{ij}(\omega) + u_{ji}(\omega) = 1 \qquad \forall i,j \in S, j > i, \forall \omega \in \Omega \tag{24}$$

$$\sum_{k\in K} y_{ik}(\omega) = 1 \qquad \forall i \in S, \forall \omega \in \Omega \tag{25}$$

$$a_j(\omega) + Y_j(\omega) + M(1-u_{ij}) \geq a_i(\omega) + Y_i(\omega) \qquad \forall i,j \in S, j \neq i, \forall \omega \in \Omega \tag{26}$$

$$a_j(\omega) + Y_j(\omega) + M(1-u_{ij}) + M(2 - y_{ik}(\omega) - y_{jk}(\omega)) \geq A_i(\omega) \qquad \forall i,j \in S, j \neq i, \forall k \in K, \forall \omega \in \Omega \tag{27}$$

$$a_j(\omega) + Y_j(\omega) + M(1-u_{ij}) \geq a_i(\omega) + Y_i(\omega) + e_i(\omega) \qquad \forall i,j \in S_r, j \neq i, \forall r \in R, \forall \omega \in \Omega \tag{28}$$

$$A_i(\omega) = a_i(\omega) + Y_i(\omega) + e_i(\omega) + W_i(\omega) \qquad \forall i \in S, \forall \omega \in \Omega \tag{29}$$

$$A_j(\omega) + M(1-u_{ij}) \geq A_i(\omega) + d_i(\omega) + q_i(\omega) \qquad \forall i,j \in S_r, j \neq i, \forall r \in R, \forall \omega \in \Omega \tag{30}$$

$$F_r(\omega) \geq A_i(\omega) + d_i(\omega) + q_i(\omega) \qquad \forall i \in S_r, \forall r \in R, \forall \omega \in \Omega \tag{31}$$

$$I_r(\omega) = F_r(\omega) - \sum_{i\in S_r} (d_i(\omega) + q_i(\omega)) \qquad \forall r \in R, \forall \omega \in \Omega \tag{32}$$

$$G_k(\omega) \geq A_i(\omega) - M(1-y_{ik}(\omega)) \qquad \forall i \in S, \forall k \in K, \forall \omega \in \Omega \tag{33}$$

$$J(\omega) = \sum_{k\in K} G_k(\omega) - \sum_{i\in S} e_i(\omega) \qquad \forall \omega \in \Omega \tag{34}$$

$$a_i(\omega) = a_i \qquad \forall i \in S,\ \forall \omega \in \Omega \tag{35}$$

$$a_i : integer \qquad \forall i \in S \tag{36}$$

$$a_i(\omega) : integer \qquad \forall i \in S, \forall \omega \in \Omega \tag{37}$$

$$u_{ij}(\omega) \in \{0,1\} \qquad \forall i,j \in S, j \neq i, \forall \omega \in \Omega \tag{38}$$

$$A_i(\omega), W_i(\omega), Y_i(\omega) \geq 0 \qquad \forall i \in S, \forall \omega \in \Omega \tag{39}$$

$$I_r(\omega), F_r(\omega), G_k(\omega), J(\omega) \geq 0 \qquad \forall r \in R, \forall k \in K, \forall \omega \in \Omega \tag{40}$$

$$y_{ik}(\omega) \in \{0,1\} \qquad \forall i \in S, \forall k \in K, \forall \omega \in \Omega \tag{41}$$

The main difference of R-SMIP from the two-stage SMIP is that the first-stage variables are defined for each scenario. However, allowing those variables to take different values in different scenarios implies that the scenario realizations in the future would be totally anticipated. Therefore, constraint (35), which maintains nonanticipativity by enforcing that $a_i(\omega)$ take the same values for each $\omega \in \Omega$, is formulated. Note that a_i variables are called as *consensus variables* in R-SMIP. Since $a_i(\omega)$ values also determine $u_{ij}(\omega)$ values through constraint (23), there is no need to add nonanticipativity constraints explicitly for $u_{ij}(\omega)$ variables.

The next step towards reaching a separable formulation requires relaxation of the nonanticipativity constraints by penalizing the violation of them in the objective function. The penalization is achieved through Lagrangian multipliers, $\mu_i(\omega), \forall i \in S, \omega \in \Omega$, and penalty parameter, ρ. Note that ordinary Euclidean norm is considered in the term where ρ is used. The formulation obtained after applying augmented Lagrangian relaxation, which we call as SQP, is as follows:

$$\begin{aligned} \min \sum_{\omega \in \Omega} p^{\omega} \Big(& c^J J(\omega) + c^I \sum_{r \in R} I_r(\omega) + c^W \sum_{i \in S} \big(W_i(\omega) + Y_i(\omega) \big) \\ & + \sum_{i \in S} \mu_i(\omega) \big(a_i(\omega) - a_i \big) + \frac{\rho}{2} \sum_{i \in S} \| a_i(\omega) - a_i \|^2 \Big) \end{aligned} \tag{42}$$

s.t.

$$(23) - (34) \tag{43}$$

$$(36) - (41) \tag{44}$$

The terms with Lagrangian multipliers and penalty components prevent separability of the formulation due to having consensus variables in them. When consensus variables are replaced by their estimated values, the obstacle can be overcome. For this purpose, a proximal point method is applied by taking weighted sum of consensus variable values for each patient over all scenarios.

The resulting parameter, which we call as *consensus parameter*, is calculated as shown below:

$$\hat{a}_i = \sum_{\omega \in \Omega} p^\omega a_i(\omega), \quad \forall i \in S \tag{45}$$

The consensus parameter must be updated at each iteration of the PHA. If the consensus parameter value is feasible for R-SMIP, then it would also represent feasible a_i values for the two-stage SMIP. The PHA aims to improve these feasible consensus parameter values over iterations and yield a near-optimal solution for the two-stage SMIP.

Replacing consensus variables in SQP with consensus parameters, we obtain the following separable stochastic quadratic formulation named SSQP.

$$\begin{aligned}
&\min \sum_{\omega \in \Omega} p^\omega \Big(c^J J(\omega) + c^I \sum_{r \in R} I_r(\omega) + c^W \sum_{i \in S} \big(W_i(\omega) + Y_i(\omega) \big) \\
&+ \sum_{i \in S} \mu_i(\omega) \big(a_i(\omega) - \hat{a}_i \big) + \frac{\rho}{2} \sum_{i \in S} \| a_i(\omega) - \hat{a}_i \|^2 \Big) \qquad (46) \\
&\text{s.t.} \\
&(23) - (34) \qquad (47) \\
&(37) - (41) \qquad (48)
\end{aligned}$$

Finally, by applying scenario decomposition on SSQP, we formulate the following scenario subproblems for each scenario $\omega \in \Omega$, called SSP, to be solved independently at each iteration of the PHA:

$$\begin{aligned}
&\min \Big(c^J J(\omega) + c^I \sum_{r \in R} I_r(\omega) + c^W \sum_{i \in S} \big(W_i(\omega) + Y_i(\omega) \big) \\
&+ \sum_{i \in S} \mu_i(\omega) \big(a_i(\omega) - \hat{a}_i \big) + \frac{\rho}{2} \sum_{i \in S} \| a_i(\omega) - \hat{a}_i \|^2 \Big) \qquad (49) \\
&\text{s.t.} \\
&a_j(\omega) \geq a_i(\omega) - M(1 - u_{ij}(\omega)) \qquad \forall i, j \in S, j \neq i \qquad (50) \\
&u_{ij}(\omega) + u_{ji}(\omega) = 1 \qquad \forall i, j \in S, j > i \qquad (51) \\
&a_i(\omega) + Y_i(\omega) = B_i(\omega) \qquad \forall i \in S \qquad (52) \\
&a_i(\omega) : integer \qquad \forall i \in S \qquad (53) \\
&u_{ij}(\omega) \in \{0, 1\} \qquad \forall i, j \in S, j \neq i \qquad (54)
\end{aligned}$$

$$(7) - (19) \tag{55}$$

4.2 Basic progressive hedging algorithm

We explain the general steps of the basic PHA, whose pseudocode is given in Algorithm 1. Letting z denote the iteration counter, we initialize z, Lagrangian multipliers and penalty parameters in step 1. Note that ρ^0 represents a nonnegative constant value. In step 2, we solve SSP models for each $\omega \in \Omega$ and determine appointment time values for each patient. However, we ignore the terms with Lagrangian multipliers and penalty parameter while solving SSP models in the first iteration. We calculate consensus parameter values for each patient by taking the weighted sum of appointment time values over scenarios in step 3. Through a positive multiplier, denoted as α, we update penalty parameter values in step 4. We update Lagrangian parameter values based on the difference between appointment time values and consensus parameter values in step 5. Finally, we check if nonanticipativity constraints are satisfied in step 6. If the constraints are satisfied, we terminate the algorithm. Otherwise, we update the iteration counter and conduct another iteration by starting from step 2.

4.3 Extended progressive hedging algorithm

Our preliminary experiments show that the basic PHA requires excessive amount of time to converge to a solution which is generally found to be of low quality. Hence, we extend the capabilities of the basic PHA by testing some enhancement ideas, and propose the algorithm that we call as *extended progressive hedging algorithm (EPHA)*.

One of the main factors that complicate the solution process is the existence of the quadratic component in the objective function of the SSP. An efficient procedure is needed to solve the SSP, as it must be solved for each scenario at each iteration of the algorithm. Therefore, we implement a linearization technique which adds cuts to each SSP independently over multiple iterations. The penalty parameter also needs a particular attention for the successful implementation of the algorithm. As it is directly represented in the objective function, and also affects Lagrangian multiplier values, the penalty parameters must be carefully updated at each iteration. Therefore, we use a dynamic penalty update method in the EPHA.

The EPHA includes an additional step in between Step 3 and Step 4 of Algorithm 1 for fixing the values of some variables. In particular, we propose a variable fixing method for the first-stage

Algorithm 1 Basic Progressive Hedging Algorithm

Step 1: Initialize parameters:
$z = 1$
$\mu_i(\omega)^{(z)} = 0 \quad \forall i \in S, \forall \omega \in \Omega$
$\rho^{(z)} = \rho^0$
Step 2: Solve scenario-subproblems:
if $z = 1$ **then**
    Solve SSP $\forall \omega \in \Omega$ by ignoring the terms with Lagrangian multipliers and penalty parameters to determine $a_i(\omega)^{(z)} \quad \forall i \in S, \forall \omega \in \Omega$
else
    Solve SSP $\forall \omega \in \Omega$ to determine $a_i(\omega)^{(z)} \quad \forall i \in S, \forall \omega \in \Omega$
Step 3: Calculate consensus parameter values:
    $\hat{a}_i = \sum_{\omega \in \Omega} p^{\omega} a_i(\omega)^{(z)} \quad \forall i \in S$
Step 4: Update penalty parameter values:
if $z > 1$ **then**
    $\rho^{(z+1)} = \alpha \rho^{(z)}$
Step 5: Update Lagrangian multiplier values:
    $\mu_i(\omega)^{(z+1)} = \mu_i(\omega)^{(z)} + \rho^{(z)}(a_i(\omega)^{(z)} - \hat{a}_i) \quad \forall i \in S, \forall \omega \in \Omega$
Step 6: Check termination criterion:
if $a_i(\omega)^{(z)} = \hat{a}_i \quad \forall i \in S, \forall \omega \in \Omega$ **then**
    Terminate the algorithm
else
    Set $z \leftarrow z + 1$
return to Step 2

variables. Finally, we use a cycle detection algorithm to prevent cyclic behaviors. The details of the EPHA are presented in the following sections.

4.3.1 Linearizing the objective function

Ignoring the penalty parameter, the quadratic component in the SSP objective function (49) can be explicitly written as follows:

$$\sum_{i\in S}\sum_{\omega\in\Omega}\|a_i(\omega)-\hat{a}_i\|^2=\sum_{i\in S}\sum_{\omega\in\Omega}a_i(\omega)^2-2\sum_{i\in S}\sum_{\omega\in\Omega}(a_i(\omega)\hat{a}_i)+|\Omega|\sum_{i\in S}\hat{a}_i^2 \tag{56}$$

The $a_i(\omega)^2$ term in the expression in (56) makes the objective function quadratic. Based on our preliminary experiments, we observe that solving a quadratic mixed-integer program for each scenario at each iteration leads to a significant computational burden. Hence, we implement a linearization method which is shown to provide a good approximation of the objective function in the studies by Helseth (2016) and Demir et al. (2021).

Letting $f(x) = a_i(\omega)^2$, the method uses the first-degree of Taylor polynomial (i.e. $f(x) \approx f(v) + f'(v)(x - v)$) to generate cuts as a lower bound on $a_i(\omega)^2$ at each iteration. The quality of approximation becomes sufficient after a certain number of iterations. Since a better approximation is obtained when v is closer to x, we set $v = a_i(\omega)^{(z-1)}$ in iteration z. In other words, the appointment time of patient i at scenario ω found in the previous iteration is considered as an appropriate point while approximating the quadratic function on the same variable in the current iteration.

In the SSP for scenario $\omega \in \Omega$ at iteration z, we replace $a_i(\omega)^2$ by a new variable, $h_i(\omega)$, and also add the following cut to the constraint set to implement the linearization method.

$$h_i(\omega) \geq (a_i(\omega)^{(z-1)})^2 + 2a_i(\omega)^{(z-1)}(a_i(\omega) - a_i(\omega)^{(z-1)}) \tag{57}$$

4.3.2 Fixing the first-stage variables

To reduce the computational time to solve SSPs and also accelerate the convergence process, we fix some first-stage variables before the termination criterion is satisfied. We determine the fixing strategies after conducting some preliminary experiments to observe the behaviors of the first-stage variables throughout the iterations.

The preliminary results show that the same precedence relationships between two patients,

$i, j \in S$, (i.e., $u_{ij}(\omega)$ values) may be observed in the vast majority of the SSP solutions in an early iteration. Furthermore, those $u_{ij}(\omega)$ values may not change over several iterations until the termination of the algorithm. Therefore, we fix $u_{ij}(\omega)$ values in the SSP for each $\omega \in \Omega$, when at least 80% of the SSPs yield the same $u_{ij}(\omega)$ value for the patients i and j at a particular iteration. When the condition is satisfied, $u_{ij}(\omega)$ is set as the value observed in the majority of the SSP solutions. As the precedence variable fixing procedure is applied over the iterations, the range for the feasible values of the appointment time variables also gets narrower.

Our observation on the behavior of the appointment time variables over iterations is similar to that for the precedence variables. Therefore, we fix $a_i(\omega)$ values for patient i when the same value, $\tilde{a}_i$, is observed in the SSP solutions across several different scenarios. The value fixed for $a_i(\omega)$ is also selected as $\tilde{a}_i$ in that case. When the appointment time for a patient is fixed, the number of variables in the SSPs decreases, and a bound is imposed on the appointment time variables for the remaining patients. Due to these effects, the appointment time fixing procedure may significantly reduce the solution time spent for solving all SSPs in a particular iteration.

Before implementing the appointment time fixing procedure, we set a threshold level that determines when the majority is achieved. The threshold level depends on the iteration counter. It starts from 100% at iteration 1, and decreases by a constant amount at each iteration until iteration $limit_2$, at which point it becomes 80%. Therefore, the amount of decrease at each iteration is set as $(20/limit_2)\%$. Next, the threshold level decreases by 10% once at iteration $limit_3$, and again at iteration $limit_4$. Note that the high threshold levels at earlier iterations prevent the risk of the convergence to poor quality solutions. On the other hand, when the number of iterations is larger, the variability in $a_i(\omega)$ values among the SSP solutions decreases, allowing us to use lower threshold levels. The details of the variable fixing procedure are presented in Algorithm 2.

To illustrate the variable fixing procedure in Algorithm 2, appointment times given by three iterations of the EPHA for a hypothetical example with three patients and six scenarios are presented in Table 2, where we set $limit_2 = 50$, $limit_3 = 60$, and $limit_4 = 70$.

Table 2(a) presents the appointment times in each scenario as a result of iteration 48. First, we calculate the $\bar{u}_{ij}^{(48)}$ values based on the equation:

$$\bar{u}_{ij}^{(z)} = \frac{1}{|\Omega|} \sum_{\omega \in \Omega} u_{ij}(\omega)^{(z)}.$$

This variable represents the fraction of scenarios where $a_i < a_j$. As an example, the appointment

Algorithm 2 Variable Fixing Procedure

if $z = 1$
  $p^{(z)} = 100$
end if
$\bar{u}_{ij}^{(z)} = \frac{1}{|\Omega|} \sum_{\omega \in \Omega} u_{ij}(\omega)^{(z)}$
for all patient pairs $(i, j), i \in S, j \in S, i \neq j$
  if $\bar{u}_{ij}^{(z)} \geq 0.8$
    Add $u_{ij}(\omega) = 1 \quad \forall \omega \in \Omega$
    Add $u_{ji}(\omega) = 0 \quad \forall \omega \in \Omega$
  end if
end for
for all patient $i, i \in S$
  if $a_{ij}(\omega)^{(z)} = \tilde{a}_i$ for $p^{(z)}\%$ of the scenarios
    Add $a_i(\omega) = \tilde{a}_i \quad \forall \omega \in \Omega$
  end if
end for
if $z \leq limit_2$
  $p^{(z+1)} = p^{(z)} - 20/limit_2$
else if $z = limit_3$
  $p^{(z+1)} = p^{(z)} - 10$
else if $z = limit_4$
  $p^{(z+1)} = p^{(z)} - 10$
end if

Table 2: Appointment times for a hypothetical example with three patients and six scenarios

(a) Iteration 48

	Patients		
Scenarios	1	2	3
1	0	27	53
2	0	15	62
3	0	20	40
4	50	0	10
5	0	15	53
6	48	22	0

(b) Iteration 65

	Patients		
Scenarios	1	2	3
1	0	20	54
2	0	15	52
3	0	20	54
4	48	0	12
5	0	15	54
6	0	22	50

(c) Iteration 72

	Patients		
Scenarios	1	2	3
1	0	22	54
2	0	14	48
3	0	22	54
4	0	22	54
5	0	18	54
6	0	22	54

time of patient 1 precedes that of patient 2 in four of the six scenarios, and thus $\bar{u}_{12}^{(48)} = 0.67$. This also implies $\bar{u}_{21}^{(48)} = 0.33$. Using similar calculations, $\bar{u}_{13}^{(48)} = 0.67$, $\bar{u}_{31}^{(48)} = 0.33$, $\bar{u}_{23}^{(48)} = 0.83$, and $\bar{u}_{32}^{(48)} = 0.17$. Among these, $\bar{u}_{23}^{(48)}$ exceeds the threshold of 0.8, and therefore u_{23} is fixed at 1 for the subsequent iterations. Similarly, u_{32} is fixed at 0. To check if any of the appointment times can be fixed, we first determine the threshold. As $48 \leq limit_2 = 50$, we use the recursive equation $p^{(z+1)} = p^{(z)} - 20/limit_2$ along with the boundary condition $p^{(1)} = 100$, implying $p^{(z)} = 100 - 20z/50 = 80.8$. Thus, if any of the patients have the same appointment time in more than 80.8% of the scenarios, their appointment time is fixed at this value. No patient in Table 2(a) has an appointment time that is identical in at least five scenarios, and hence none of the appointment times is fixed.

Table 2(b) shows the appointment times in iteration 65. As expected due to having fixed $u_{23} = 1$ and $u_{32} = 0$, $a_2 < a_3$ in all scenarios. We also calculate $\bar{u}_{12}^{(65)} = \bar{u}_{13}^{(65)} = 0.83$ and $\bar{u}_{21}^{(65)} = \bar{u}_{31}^{(65)} = 0.17$. Since both $\bar{u}_{12}^{(65)}$ and $\bar{u}_{13}^{(65)}$ both exceed the threshold of 0.8, we fix $u_{12} = u_{13} = 1$ and $u_{21} = u_{31} = 0$ for subsequent iterations. For appointment times, as $limit_3 \leq 65 < limit_4$, the threshold is calculated from $p^{(z)} = p^{(60)} - 10 = 70$. Hence, if a patient's appointment times are identical in at least 70% of the scenarios, they can be fixed. We observe this to be the case for patient 1, where $a_1 = 0$ in five scenarios. Consequently, we fix the appointment time of this patient at $t = 0$.

The appointment times in iteration 72 are given in Table 2(c). Due to the fixing in iteration 65, all scenarios follow the same order (1-2-3) of appointment times for the patients. Similarly, $a_1 = 0$ holds for all scenarios. Due to $72 \geq limit_4$, the threshold to fix appointment times in this iteration is given by $p^{(z)} = p^{(70)} - 10 = 60$. As $a_2 = 22$ in four out of six scenarios and $a_3 = 54$ in five of the six scenarios, these values can be fixed. This implies that by this stage, all appointment time values have converged and the EPHA can terminate.

During the implementation of the basic PHA, the appointment time values of all patients in the SSPs for all scenarios may stay the same over consecutive iterations. To prevent such a cyclic behavior and the risk of non-convergence, we implement the cycle detection approach proposed by Watson and Woodruff (2011). The approach detects cycles by tracking the Lagrangian multipliers for each patient, whose values change even if a cycle occurs. When a cycle is detected, we fix the appointment time value to the $\hat{a}_i$ value found at that iteration. Note that the cycle detection is implemented at each iteration after iteration $limit_2$.

The cycle detection and variable fixing procedures we implement ensure the convergence of the EPHA. However, even though it occurs rarely, the EPHA may run through an extensive number of

iterations in some instances, because the appointment times of a number of patients can not be fixed. We observe in our preliminary experiments that after a certain iteration (we call it as $limit_5$), the unfixed appointment times change slightly. Therefore, to prevent running the algorithm for several minutes only to gain a minor improvement in the objective value, we review the unfixed patient appointment times every *controliter* iterations after iteration $limit_5$. If the number of unfixed appointment times does not change in the last *controliter* iterations, we fix the appointment time of each patient i whose appointment time was not fixed. Note that the fixed value is chosen as the $\hat{a}_i$ value in that iteration. For the example in Table 2(c), let $limit_5 = 90$ and $controliter = 100$, and suppose that the a_2 and a_3 values are not fixed due to different thresholds than those in the example. If these appointment time values persisted for all scenarios from iterations 90 to 190, we fix a_2 and a_3 as the average over all scenarios, namely at 20 and 53, respectively.

4.3.3 Penalty Parameter Update Method

The selection of penalty parameter values is critical to ensure that the EPHA converges to a good quality solution in a reasonable amount of time. Setting very high values for ρ enforces the convergence of appointment time values in the SSPs for different scenarios to the related consensus parameter values in the early iterations. However, the consensus parameter values are estimated progressively, and their estimations can be of low quality in earlier iterations. Note that the initial estimation of consensus parameters are obtained by solving SSPs without any non-anticipativity constraints. Therefore, very high values of ρ may result in low quality model solutions. On the other hand, very low ρ values would delay the convergence of the appointment time values, and hence increase the number of iterations. As the number of iterations increases, the appointment time values can be better estimated at the expense of increased computational time. Hence, the EPHA can converge to a good quality solution after extensive number of iterations. Therefore, the value of ρ must be set carefully at each iteration by examining its relationship with the solution quality and run time.

The previous research shows that using a dynamic update method to change parameter values provide significant advantages in terms of solution quality and run time (Hvattum and Løkketangen 2009, Gul et al. 2015, Demir et al. 2021). We enhance the penalty update method suggested by Demir et al. (2021), which is an extended version of Hvattum and Løkketangen (2009).

The method uses two parameters that help examine the convergence pattern of both primal and dual solutions. The first parameter, denoted by Δ_d, represents the squared sum of the differences

between appointment time values in the SSP solution for each scenario and the relevant consensus parameter. This parameter provides information on the convergence level at a particular iteration. If Δ_d increases from one iteration to another, this means the SSP solutions are moving away from the consensus condition. The second parameter, denoted by Δ_p, considers the squared sum of the difference between the consensus parameter value in the current iteration and that in the previous iteration, and sum the differences over all patients. An increase in Δ_p implies that the EPHA is moving from the previous target to another target of convergence.

The method first checks the change in Δ_d value with respect to that of the previous iteration. If the change is positive, ρ is multiplied by $\alpha > 1$ to ease convergence. Otherwise, it implies that the EPHA solution is approaching a consensus point. Next, the method checks the change in Δ_p value from the previous iteration to the current one. If the change is positive, ρ is multiplied by $\frac{1}{\alpha}$ to prevent the convergence to a poor solution. Otherwise, the method keeps ρ stable, since the decrease in Δ_p value means that the oscillation from one consensus point to another one is fading.

We impose a three-phase limit on ρ while implementing the penalty update method to reduce the risk of immature termination of the EPHA, which may occur due to the following reasons: (i) the number of cuts added to approximate the quadratic term becomes sufficient only after a certain number of iterations, or (ii) a rapid increase in ρ may activate variable fixing procedures unnecessarily. Until iteration $limit_1$, ρ can not exceed ρ^{u1}. After that point, the upper limit is updated to ρ^{u2}, which is greater than ρ^{u1}. The second upper limit is valid until iteration $limit_5$. Afterwards, the upper limit is updated to ρ^{u3}, a value greater than ρ^{u2}. Note that $limit_5$ is the threshold level also used within the variable fixing procedure. The details of the penalty parameter update method is shown in Algorithm 3.

To illustrate the working principles of the penalty parameter update method, Table 3 shows the appointment times resulting from two subsequent iterations of the EPHA for a hypothetical example with three patients and four scenarios. Suppose we set $limit_1 = 25$, $limit_5 = 90$, $\rho^{u1} = 0.01$, $\rho^{u2} = 0.05$, $\rho^{u3} = 0.1$, and $\alpha = 1.5$. Suppose further that the average appointment times in iteration 54 are $\hat{a}_1^{(54)} = 14$, $\hat{a}_2^{(54)} = 18.5$, and $\hat{a}_3^{(54)} = 37.5$, and that the resulting penalty parameter at the end of iteration 55 is $\rho^{(56)} = 0.0369$. The algorithm is currently in iteration 56.

From Table 3(a), we calculate $\hat{a}_1^{(55)} = (0 + 0 + 0 + 50)/4 = 12.5$. Similarly, $\hat{a}_2^{(55)} = 15.5$ and $\hat{a}_3^{(55)} = 41.5$. We also have $\Delta_d^{(55)} = \sum_{i \in S} \sum_{\omega \in \Omega} (a_i(\omega)^{(55)} - \hat{a}_i^{(55)})^2 = (0-12.5)^2 + (0-12.5)^2 + \ldots + (10-41.5)^2 = 3839$ and $\Delta_p^{(55)} = \sum_{i \in S} (\hat{a}_i^{(55)} - \hat{a}_i^{(54)})^2 = (12.5-14)^2 + (15.5-18.5)^2 + (41.5-37.5)^2 = 27.25$.

Algorithm 3 Penalty Update Method with Three Upper Limits

$\Delta_d^{(z)} = \sum_{i \in S} \sum_{\omega \in \Omega} (a_i(\omega)^{(z)} - \hat{a}_i^{(z)})^2$
$\Delta_p^{(z)} = \sum_{i \in S} (\hat{a}_i^{(z)} - \hat{a}_i^{(z-1)})^2$
if $z \leq limit_1$
    $\rho^u = \rho^{u1}$
else if $z \leq limit_5$
    $\rho^u = \rho^{u2}$
else
    $\rho^u = \rho^{u3}$
if $\Delta_d^{(z)} - \Delta_d^{(z-1)} > 0 \quad \& \quad \rho^{(z)} < \rho^u$
    $\rho^{(z+1)} = \alpha \rho^{(z)}$
else if $\Delta_d^{(z)} - \Delta_d^{(z-1)} > 0$
    $\rho^{(z+1)} = \rho^u$
else if $\Delta_p^{(z)} - \Delta_p^{(z-1)} > 0$
    $\rho^{(z+1)} = \frac{1}{\alpha} \rho^{(z)}$
else if $\rho^{(z)} > \rho^u$
    $\rho^{(z+1)} = \rho^u$
else
    $\rho^{(z+1)} = \rho^{(z)}$

Table 3: Appointment times in two subsequent iterations for a hypothetical example with three patients and four scenarios

(a) Iteration 55

	Patients		
Scenarios	1	2	3
1	0	27	54
2	0	15	62
3	0	20	40
4	50	0	10

(b) Iteration 56 (current)

	Patients		
Scenarios	1	2	3
1	0	25	55
2	0	15	62
3	0	18	42
4	0	12	45

Carrying on the calculations for iteration 56, we obtain $\hat{a}_1^{(56)} = 0$, $\hat{a}_2^{(56)} = 17.5$, $\hat{a}_3^{(56)} = 51$, $\Delta_d^{(56)} = 347$, and $\Delta_p^{(56)} = 250.5$. As $limit_1 < 56 < limit_5$, $\rho^u = \rho^{u2} = 0.05$. We also have $\Delta_d^{(56)} - \Delta_d^{(55)} = 347 - 3389 < 0$ and $\Delta_p^{(56)} - \Delta_p^{(55)} = 250.5 - 27.25 > 0$. Hence, the algorithm is approaching a consensus point, but there is a possibility of approaching a poor solution. Therefore, the penalty parameter should be decreased to avoid this convergence. Altogether, these conditions lead to the update equation $\rho^{(57)} = \frac{1}{\alpha}\rho^{(56)} = \frac{0.0369}{1.5} = 0.0246$.

5 Experimental Study

We test the EPHA using the data set of an outpatient procedure center in Mayo Clinic (Gul et al. 2011). We create problem instances based on the surgeries performed at the urology department. We sample induction and surgery durations from the data set which includes the records of 1963 patients whose surgeries were performed over the data collection period. Urology surgeries are grouped into five acuity levels in the data set. The number of surgeries of each acuity level and descriptive statistics related to induction and surgery durations are given in Table 4. We sample induction and surgery durations independently for each acuity level. We generate turnover durations by assuming a uniform distribution with lower and upper limits as 15 and 30 minutes, respectively. We implement the algorithms in Microsoft Visual C++ 2019 using CPLEX 12.8 Concert Technology. We conduct the experiments on an Intel(R) Xeon(R) E-2246G computer with six-core processor running at 3.60 GHz and 16GB RAM.

Each experiment is conducted on an instance set that consists of 10 instances. An instance set represents the case where 7 patients are scheduled for 2 IRs and 3 ORs. We ensure that occurrence frequencies of acuity levels in a daily patient list are proportional to the patient count values for each acuity level in the data set given in Table 4. The instances in a given instance set differ from each other based on the induction and surgery durations. In other words, different set of induction and surgery duration values are sampled for a patient for each instance. The number of duration values sampled for each patient in an instance is set to 50 (i.e. $|\Omega| = 50$) in all experiments, except where we assess the optimality gap.

We consider the trade-off between idle time and waiting time by setting different values for the cost coefficients, c^I, c^J, c^W. Instead of setting actual cost values for the coefficients, we use them as trade-off parameters whose values add up to 1 in each experiment.

We set the values of the parameters used within the variable fixing and penalty update methods

Table 4: Number of patients, mean and standard deviations of induction and surgery durations (in minutes) for each acuity level

		Induction		Surgery	
Acuity level	**Count**	**Mean**	**Std. deviation**	**Mean**	**Std. deviation**
1	329	23.65	5.66	29.65	20.52
2	640	13.82	6.07	17.48	8.58
3	153	29.03	6.78	109.12	42.96
4	345	24.97	5.67	30.88	14.12
5	496	28.20	7.42	52.12	32.56

Table 5: Benefit of variable fixing procedure in terms of objective value and run time (in seconds)

	Objective Value		Run Time	
Instance #	**Fixing**	**No Fixing**	**Fixing**	**No Fixing**
1	73.27	79.58	796.20	10800.00
2	71.76	75.40	1153.85	10800.00
3	70.77	78.37	715.11	10800.00
4	67.36	67.44	645.28	10800.00
5	75.91	75.87	678.02	10800.00
6	77.40	78.59	852.65	10800.00
7	70.41	71.28	891.20	10800.00
8	76.27	93.76	715.64	10800.00
9	72.25	77.82	923.53	10800.00
10	75.03	79.86	893.79	10800.00
Average	73.04	77.79	826.53	10800.00

based on our preliminary experiments as $limit_1 = 25$, $limit_2 = 50$, $limit_3 = 60$, $limit_4 = 70$ and $limit_5 = 90$, $controliter = 100$.

5.1 Benefit of variable fixing and penalty update methods

We assess the value of implementing variable fixing procedure and penalty update method. We set c^I, c^J, c^W as 0.5, 0.25, 0.25 in these experiments, respectively. Table 5 compares the performance of the algorithm with and without variable fixing procedure over 10 instances. The results clearly illustrate the significant benefit of the variable fixing procedure. The table shows that when the variable fixing mechanism is implemented, the objective value improves by 6.1% on average. Furthermore, the algorithm terminates due to time limit of 3 hours if the variable fixing procedure is not incorporated into the algorithm. On the other hand, the EPHA finds the solutions in around 14 minutes when the variable fixing procedure is implemented.

We next show the benefit of implementing the penalty update method within the EPHA. As mentioned in Section 4, our penalty update method considers 3 upper limits ($\rho^{u1}, \rho^{u2}, \rho^{u3}$). To illustrate the benefit of using multiple upper limits, we first conduct experiments using the method

Table 6: Benefit of the penalty update method (using 1 or 2 upper limits) in terms of objective value and run time (in seconds)

	Objective Value		**Run Time**	
Instance #	**2**	**1**	**2**	**1**
1	73.27	73.57	796.20	2894.29
2	71.76	71.73	1153.85	1443.02
3	71.77	71.44	715.11	1024.66
4	67.36	67.46	645.28	955.09
5	75.91	75.85	678.02	931.84
6	77.40	77.45	852.6	929.56
7	70.41	70.33	891.20	1020.38
8	76.27	76.53	715.64	1811.83
9	72.25	72.21	923.53	885.00
10	75.03	74.45	893.79	1169.84
Average	73.04	73.10	826.53	1306.55

Table 7: Benefit of the penalty update method (using 2 or 3 upper limits) in terms of objective value and run time (in seconds)

	Objective Value		**Run Time**	
Instance #	**3**	**2**	**3**	**2**
1	78.05	78.11	1380.02	1488.61
2	80.97	80.99	1390.41	1677.11
3	79.07	79.19	1266.22	1433.48
4	76.57	76.34	1556.31	2367.65
5	81.98	81.98	1674.81	1531.63
6	78.16	78.53	1599.59	2327.15
7	79.94	79.94	1614.65	1619.08
8	84.61	98.52	3324.43	6171.85
9	80.99	81.23	1674.25	2072.15
10	72.86	72.56	1678.84	2630.24
Average	79.32	80.74	1715.95	2331.88

with 1 (ρ^{u1}) and 2 upper limits (ρ^{u1}, ρ^{u2}) based on the baseline instances. Table 6 shows that when the number of imposed upper limits increases from 1 to 2, the run time improves significantly (by 27%), while the objective value improves by 0.08%. We then conduct tests using 2 (ρ^{u1}, ρ^{u2}) and 3 upper limits ($\rho^{u1}, \rho^{u2}, \rho^{u3}$) on another instance set selected to ensure that the EPHA iteration counter exceeds $limit_5$ such that the limit ρ^{u3} can be imposed in all instances. Table 7 illustrates that increasing the number of upper limits from 2 to 3 provides a greater improvement in terms of the objective value (by 1.76%). Since the run time also improves by 26%, we continue with using 3 upper limits while implementing the penalty update method in the remainder of the experiments.

Table 8: Comparison of the EPHA with CPLEX with respect to the objective values and run times (in seconds) for each instance set

		Objective Value			Run Time	
Set #	c^I, c^J, c^w	**EPHA**	**CPLEX**	**Difference (%)**	**EPHA**	**CPLEX**
1	0.33, 0.33, 0.34	59.05	56.2	4.73	123.86	2317.97
2	0.5, 0.25, 0.25	67.27	66.55	1.05	71.12	3236.03
3	0.65, 0.10, 0.25	73.92	72.91	1.39	88.47	1844.64
Average		66.74	65.22	2.39	94.48	2466.19

5.2 Comparison of the EPHA results with optimal solutions

We assess the performance of the EPHA by comparing the algorithm with CPLEX on three different instance sets. The sets are differentiated from each other by varying cost values (i.e., the number of surgeries from each acuity level does not change). Since resource utilization is given higher priority over patient satisfaction by the decision makers in hospitals, we ensure that the unit cost of waiting time does not exceed the total unit costs of OR and IR idle times in these experiments. Furthermore, we consider only the realistic cases where the unit cost of IR idle time does not exceed that of OR idle time. Note that all other cases are considered in Section 5.4. In particular, we consider the following values for the triplet, c^I, c^J, c^W: (0.33, 0.33, 0.34),(0.5, 0.25, 0.25), (0.65, 0.10, 0.25). We set $|\Omega| = 10$ in these experiments, as CPLEX cannot find optimal solutions for larger values of scenario sizes.

Table 8 shows that the average gap between the EPHA and CPLEX over all instances is 2.39%, while it can be as low as 1.05% for an instance set. The EPHA finds the solution in about 1.5 minutes on average, while the CPLEX requires more than 41 minutes to obtain the optimal solution.

5.3 Comparison of the EPHA with scheduling heuristics

We compare the EPHA with the commonly used scheduling heuristics in the earlier literature on surgery or chemotherapy appointment scheduling (Gul et al. 2011, Gul 2018, Demir et al. 2021). In particular, we make comparisons with combinations of sequencing and job hedging heuristics.

Using sequencing heuristics, we sort the surgeries based on one of the three rules: (1) smallest expected induction duration first (SPT), (2) largest expected induction duration first (LPT), and (3) smallest variance of induction first (VAR). Note that we choose induction duration as the criterion, since the sequencing variables in the two-stage SMIP model exist only for the induction activity. Furthermore, as can be verified by Table 4, SPT and LPT sequences do not change even if the patients are sequenced based on surgery duration or total duration. The sequence for VAR

Table 9: Comparison of EPHA with scheduling heuristics

	Difference (%)		
Percentile	**SPT**	**LPT**	**VAR**
50%	42.33	71.93	54.38
60%	40.29	70.66	54.25
70%	39.20	70.22	53.76
80%	37.89	70.73	52.04
90%	37.52	66.72	54.76
Average	39.45	70.05	53.84

rule may change in such a case, but our preliminary experiments show that the induction duration is a better criterion.

After sequencing the surgeries, we calculate the appointment times of patients. Initially, the appointment times of the first patients of each IR are set as 0 (i.e., the beginning of the shift). We then identify the patient whose induction is expected to finish the earliest among all patients in the IR based on their induction durations, which are estimated based on the job hedging procedure. Let that patient be indexed by i. Next, the first patient among the non-treated patients is identified (let index j denote that patient). The appointment time of patient j is then set by adding the estimated induction duration of patient i to the appointment time of patient i. In other words, the appointment time of patient j is set equal to the planned induction finish time of patient i. Due to the trade-off between waiting time and idle time, the estimation of induction duration is critical. We estimate the induction duration for each acuity level separately, while applying job hedging. Job hedging procedure allows to use different percentiles of the relevant distribution while estimating the induction duration. In our experiments, we test 50^{th} (median), 60^{th}, 70^{th}, 80^{th}, and 90^{th} percentiles. Instead of fitting general distributions, we sort the induction durations for each acuity level in our data set, and identify the values corresponding to the tested percentiles based on the sample. When, the appointment time of patient j is set, the next patient among the non-treated patients is identified similarly. We set the appointment time of this patient, and then follow the same procedure, iteratively, to set the appointment times of all patients in the daily list.

We set c^I, c^J, c^W as 0.5, 0.25, 0.25 in these experiments, respectively. The average EPHA objective value over 10 instances in the set is found as 73.04. Table 9 provides the percentage improvement that the EPHA solution provides over the heuristic solutions with respect to the objective value. Among the sequencing heuristics, SPT performs the best on average. However, the EPHA improves SPT solutions by almost 40%. The improvement over VAR and LPT solutions are around 54% and 70%, respectively. Note that increasing job hedging level results in better

Table 10: Average objective values, run times (in seconds), idle times and patient waiting times (in minutes) of EPHA for different unit costs of the three objectives

c^I	c^J	c^W	**Objective**	**Run time**	**OR idle time**	**IR idle time**	**Waiting time**
0.833	0.083	0.083	78.14	626.46	79.17	63.41	82.60
0.500	0.250	0.250	72.51	907.00	92.17	49.25	56.47
0.476	0.476	0.048	62.90	699.92	86.37	35.65	100.68
0.476	0.048	0.476	69.27	807.75	100.47	61.52	38.83
0.400	0.400	0.200	64.39	1003.45	98.61	37.93	48.87
0.400	0.200	0.400	65.58	1131.80	102.45	44.33	39.33
0.250	0.500	0.250	56.96	1023.33	128.20	25.41	48.82
0.250	0.250	0.500	58.08	1240.47	130.73	44.05	28.78
0.200	0.400	0.400	53.04	1189.69	134.61	32.07	33.21
0.083	0.833	0.083	35.84	876.44	228.88	13.15	69.76
0.083	0.083	0.833	28.92	1334.53	178.77	65.73	10.25
0.048	0.476	0.476	31.31	1134.13	162.15	24.51	25.03

results when SPT is used to sequence patients, but we cannot generalize this observation to other sequencing rules. The overall results indicate that it is highly essential to use a sophisticated algorithm such as the EPHA to schedule surgeries for surgical suites that follow parallel processing principle.

5.4 Sensitivity analysis on the model and problem parameters

In this section, we present our experiments on analyzing the sensitivity of the results of EPHA with respect to changing unit costs of the three objectives and the number of induction rooms.

5.4.1 Impact of the unit costs of the three objectives

Our first sensitivity analysis covers the case where we vary the relative unit costs of OR idle time, IR idle time and patient waiting time to observe the trade-off between these three objectives. For this end, we use 12 different combinations of c^I, c^J and c^W by setting two of the costs equal to each other and the remaining third to be 10%, 50%, 200%, and 1000% of the former two. We then normalize these unit costs to ensure that they add up to 1.

For each of the 12 unit cost combinations, Table 10 provides the average objective function value, computational time, OR idle time, IR idle time, and patient waiting time for each instance set. Focusing on each of the last three columns yields insights into the sensitivity of each of these objectives to the change in unit costs. Among these, OR idle time ranges from 79.17 in the case where its unit cost is 10 times that of the other two objectives to 228.88 minutes when the cost per minute of IR idle time is 10 times that of OR idle time and patient waiting time. Similarly, IR

Table 11: Average objective values, run times (in seconds), idle times and patient waiting times (in minutes) of EPHA for varying number of induction rooms

Number of IRs	Objective	Run Time	OR Idle Time	IR Idle Time	Waiting Time
1	129.14	802.21	187.42	53.49	88.23
2	72.51	907.00	92.17	49.25	56.47
3	67.68	1707.79	83.42	55.19	48.70

idle time ranges from 13.15 to 63.41 minutes, and patient waiting time varies from 10.25 to 100.68 minutes. Whereas the range is largest for OR idle time, patient waiting time sees the highest relative increase from average, with a value twice as long in the worst case. The variations of all three objectives with regard to unit cost changes underlines the significance of calculating these costs accurately when determining the surgery schedules.

When a pairwise analysis of changes is made between the objectives, one observes a negative correlation between each pair, which shows that these three objectives are in total conflict with one another, pointing to the importance of incorporating all three objectives into the model. The most significant trade-off occurs between OR and IR idle time, with a correlation coefficient of -0.44. The corresponding coefficients for (i) OR idle time and waiting time, and (ii) IR idle time and waiting time are -0.30 and -0.28, respectively.

5.4.2 Impact of the number of induction rooms

An important decision for surgery scheduling in the existence of parallel processing is the number of IRs in the system. While a large number of IRs may improve the performance measures, it has been observed in the literature that using more IRs leads to the need to hire more anesthesia and nursing personnel (Harders et al. 2006). If the management decides to staff additional IRs with the existing personnel, this would induce opportunity cost, as the personnel could be utilized elsewhere in the hospital. Hence, care should be taken into consideration in determining the number of IRs. To analyze the effect of the number of IRs on system performance, we extend our experiments to involve (i) a single IR and (ii) three IRs, in addition to the two-IR case in our baseline setting.

Table 11 summarizes the results with varying numbers of IRs. The results show that adding a second IR decreases the overall objective by around 45%, whereas the benefit of adding the third IR is less than 7%. As expected, the law of diminishing returns applies in this case. We observe similar outcomes when comparing the OR idle times and patient waiting times for the three settings, where the reduction is more substantial with the addition of the second IR, but decreases significantly with the third one. IR idle time does not change substantially with increasing number of IRs.

Table 12: Patient waiting times and OR closure times (in minutes) of EPHA under different parallel and serial processing settings

	2IR-parallel			**3IR-parallel**			**Serial**		
$q_i(\omega)/e_i(\omega)$	0.5	1	2	0.5	1	2	0.5	1	2
Waiting time	51.85	58.05	67.34	47.74	53.33	67.49	36.12	39.65	52.78
OR1 closure	92.31	107.36	144.61	85.68	103.33	141.48	109.07	127.49	167.30
OR2 closure	153.54	178.87	237.19	143.46	168.66	227.59	166.85	193.78	258.12
OR3 closure	252.02	285.41	361.02	246.65	286.57	349.74	301.78	340.29	413.22

While this may not be an intuitive result, it is due to the way idle time is calculated based on the IR closure times. As more IRs are added, they become less utilised. However, at the same time, closure times are shortened, offsetting the increase in the idle time and resulting in similar numbers for all settings.

5.5 Comparison of parallel processing and serial processing

As opposed to a serial system where both induction and turnover are performed within the OR, the parallel processing approach allows performing these activities simultaneously in different rooms, increasing the utilization of ORs. The main advantage with a serial system would be a decrease in patient waiting times, as a parallel system involves both waiting for the IR and the OR, whereas a serial system only includes waiting for the OR. On the other hand, in a serial system, the need for carrying out induction and turnovers in the same OR implies that the room will be closed at a later time compared to a parallel processing system.

To compare the parallel processing approach against an equivalent serial system, we consider two parallel systems, one with two and one with three IRs, as well as a serial system without an IR. In all three cases, there are three ORs. We also vary the ratio between the turnover and induction time as 0.5, 1, and 2 to observe whether the conclusions change with regard to the durations of these activities. As the serial system does not involve any IRs, IR idle time cannot be used as a performance measure. Due to the same reason, OR idle time ceases to be a fair comparison measure. Consequently, we use the closure times of the three ORs, in addition to the total patient waiting time, to compare these two systems. The results of our experiments are summarized in Table 12.

Table 12 shows the trade-off between patient waiting time and OR closure times between the two systems quite clearly. The serial system yields up to 20 and 15 minutes of total patient waiting time savings over a 2IR and a 3IR system, respectively. On the other hand, these savings are more than offset in the parallel processing system by the savings in OR closure times. With two IRs,

Table 13: Average objective values for the mean value solution (MV), EPHA solution, and relative VSS for the instances

Instance	MV	EPHA	%VSS
1	78.40	74.20	5.67%
2	77.74	73.71	5.46%
3	82.92	76.06	9.03%
4	75.76	70.17	7.97%
5	75.64	70.46	7.36%
6	77.83	74.31	4.74%
7	77.38	73.77	4.89%
8	77.19	70.83	8.97%
9	84.97	80.21	5.93%
10	73.24	69.11	5.98%

the total savings varies between 80 and 95 minutes. Setting up three IRs increases the savings to between 100 and 120 minutes. The findings are robust over varying ratios of the ratio of turnover to induction time. These findings underline the increase in overall efficiency when using a parallel processing system.

5.6 Estimation of the Value of Stochastic Solution

Our last set of experiments focus on estimating the value of stochastic solution (VSS). VSS aims to assess the additional benefits obtained by considering the uncertainty of the problem environment when determining the decisions in the first stage of the problem. To estimate the VSS, we first solve the mean value problem. This is obtained by setting all the uncertain parameters at their expected values and solving a deterministic model. The objective values arising from the first-stage decisions of the mean value problem are compared against those of the EPHA to calculate the relative VSS values, given in Table 13.

For all of our 10 instances generated based on the baseline parameter settings, the relative VSS ranges between 5% and 10%. While this may not appear to be a considerable change, it points to a substantial savings in the OR costs. The findings from Table 13 underline the importance of considering the uncertainty in the surgery, induction and turnover times when deciding on the appointment times of the patients for a surgical suite following parallel processing principles.

6 Conclusions

In this study, we propose a two-stage SMIP model for the problem of scheduling surgeries for multiple ORs and IRs that function based on parallel processing principle. The parallel processes

in our problem refer to the concurrent implementation of induction in an IR and turnover in an OR. We consider the uncertainty in induction, surgery and turnover durations while making scheduling decisions. Using the two-stage SMIP model, we first sequence patients and set appointment times for surgeries. After the realization of uncertain durations, we assign patients to IRs independently for each scenario. We show that the special structure of the model allows the use of an optimal myopic policy for IR assignment decisions. We minimize the expected total cost of patient waiting time, OR idle time and IR idle time, whose values are calculated at the second stage of the two-stage SMIP model. We add valid inequalities by imposing bounds on variables and formulating symmetry-breaking constraints to enhance the model formulation.

We implement an extended version of the PHA, which we call as EPHA, to solve the two-stage SMIP model. We propose a penalty update method and a variable fixing mechanism within the EPHA. Besides, we add cuts to linearize the quadratic objective function of the scenario subproblems and implement a cycle detection mechanism using the procedures proposed in the earlier literature.

We illustrate the advantages provided by the EPHA over the basic PHA by conducting a comprehensive set of experiments using the data set of a large academic hospital in the US. We compare the EPHA with CPLEX and show that the algorithm provides near-optimal solutions within a reasonable amount of time. We also compare the EPHA with several combinations of simple sequencing and job hedging heuristics and indicate that a sophisticated solution approach is necessary to solve the two-stage SMIP model. We assess the VSS to emphasize the importance of considering uncertainty in induction, surgery and turnover durations.

Based on the results obtained through near-optimal schedules, we show that a parallel processing system significantly outperforms the serial system in terms of OR closure times. The finding is consistent with the results of the actual trials conducted in the hospitals without optimizing surgery schedules. The lower OR closure times in parallel processing systems allows a surgical suite manager to increase daily throughput. On the other hand, the average total patient waiting time for a parallel processing system is found to be worse than that of the serial system. Therefore, the manager must carefully consider the trade-off between patient satisfaction and daily throughput while choosing a particular setting for the flow of patients. As the number of IRs increases, the manager can better benefit from the parallel processing principle. However, the additional benefits must be weighed against the additional staffing costs while determining the number of IRs.

We consider 7 patients in our experiments, because we obtain reasonable performance measure

values in that case for the baseline setting of our study (2 IRs, 3ORs). The EPHA can solve instances up to 13 patients under the same setting in 3 hours, but can be insufficient for larger surgical suites where a greater number of ORs are reserved for surgical groups. We plan to handle those cases in a future study by decreasing the computational time through parallelizing the scenario subproblem solution process of the EPHA. Furthermore, we make the assumption that the patient-to-OR assignment decisions are made in advance in this study. We plan to relax this assumption and create a model for a more flexible case in a future study.

References

Ahmadi-Javid A, Jalali Z, Klassen KJ (2017) Outpatient appointment systems in healthcare: A review of optimization studies. *European Journal of Operational Research* 258(1):3–34.

Atighehchian A, Sepehri MM, Shadpour P, Kianfar K (2020) A two-step stochastic approach for operating rooms scheduling in multi-resource environment. *Annals of Operations Research* 292:191–214.

Bai M, Storer R, Tonkay G (2017) A sample gradient-based algorithm for a multiple-or and pacu surgery scheduling problem. *IISE Transactions* 49(4):367–380.

Batun S, Denton BT, Huschka TR, Schaefer AJ (2011) Operating room pooling and parallel surgery processing under uncertainty. *INFORMS journal on Computing* 23(2):220–237.

Birge JR, Louveaux F (2011) *Introduction to stochastic programming* (Springer Science & Business Media).

Cayirli T, Veral E (2003) Outpatient scheduling in health care: a review of literature. *Production and Operations Management* 12(4):519–549.

Crainic TG, Fu X, Gendreau M, Rei W, Wallace SW (2011) Progressive hedging-based metaheuristics for stochastic network design. *Networks* 58(2):114–124.

Demir NB, Gul S, Çelik M (2021) A stochastic programming approach for chemotherapy appointment scheduling. *Naval Research Logistics (NRL)* 68(1):112–133.

Denton B, Gupta D (2003) A sequential bounding approach for optimal appointment scheduling. *IIE Transactions* 35(11):1003–1016.

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

El-Boghdadly K, Nair G, Pawa A, Onwochei DN (2020) Impact of parallel processing of regional anesthesia with block rooms on resource utilization and clinical outcomes: a systematic review and meta-analysis. *Regional Anesthesia and Pain Medicine* 45(9):720–726.

Erdogan S, Denton B (2013) Dynamic appointment scheduling of a stochastic server with uncertain demand. *INFORMS Journal on Computing* 25(1):116–132.

Friedman DM, Sokal SM, Chang Y, Berger DL (2006) Increasing operating room efficiency through parallel processing. *Annals of Surgery* 243:10–14.

Gonçalves RE, Finardi EC, da Silva EL (2012) Applying different decomposition schemes using the progressive hedging algorithm to the operation planning problem of a hydrothermal system. *Electric Power Systems Research* 83(1):19–27.

Gul S (2018) A stochastic programming approach for appointment scheduling under limited availability of surgery turnover teams. *Service Science* 10(3):277–288.

Gul S, Denton BT, Fowler JW (2015) A progressive hedging approach for surgery planning under uncertainty. *INFORMS Journal on Computing* 27(4):755–772.

Gul S, Denton BT, Fowler JW, Huschka T (2011) Bi-criteria scheduling of surgical services for an outpatient procedure center. *Production and Operations Management* 20(3):406–417.

Gupta D, Denton B (2008) Appointment scheduling in health care: Challenges and opportunities. *IIE transactions* 40(9):800–819.

Hanss R, Buttgereit B, Tonner P, Bein B, Schleppers A, Steinfath M, Scholz J, Bauer M (2005) Overlapping induction of anesthesia. *Anesthesiology* 103:391–400.

Harders M, Melangoni M, Weight S, Sidhu T (2006) Improving operating room efficiency through process redesign. *Anesthesiology* 140(4):509–516.

Helseth A (2016) Stochastic network constrained hydro-thermal scheduling using a linearized progressive hedging algorithm. *Energy Systems* 7(4):585–600.

Hvattum LM, Løkketangen A (2009) Using scenario trees and progressive hedging for stochastic inventory routing problems. *Journal of Heuristics* 15(6):527.

Khaniyev T, Kayis E, Gullu R (2020) Next-day operating room scheduling with uncertain surgery durations: Exact analysis and heuristics. *European Journal of Operational Research* 286(1):49–62.

Laporte G, Louveaux FV (1993) The integer l-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters* 13(3):133–142.

Lee S, Yih Y (2014) Reducing patient-flow delays in surgical suites through determining start-times of surgical cases. *European Journal of Operational Research* 238:620–629.

Mancilla C, Storer R (2012) A sample average approximation approach to stochastic appointment sequencing and scheduling. *IIE Transactions* 44(8):655–670.

Mancilla C, Storer R (2013) Stochastic sequencing of surgeries for a single surgeon operating in parallel operating rooms. *IIE Transactions on Healthcare Systems Engineering* 3(2):127–138.

Marjamaa R, Torkki P, Hirvensalo E, Kirvela O (2009) What is the best workflow for an operating room? a simulation study of five scenarios. *Health Care Management Science* 12:142–146.

Rockafellar RT, Wets RJB (1991) Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research* 16(1):119–147.

Sen S, Higle JL (2005) The c3 theorem and a d2 algorithm for large scale stochastic mixed-integer programming: set convexification. *Mathematical Programming* 104(1):1–20.

Smith MP, Sandberg WS, Foss J, Massoli K, Kanda M, Barsoum W, Schubert A (2006) High-throughput operating room system for joint arthroplasties durably outperforms routine processes. *Anesthesiology* 109:25–35.

Torkki P, Marjamaa R, Torkki M, Kallio P, Kirvela O (2005) Use of anesthesia induction rooms can increase the number of urgent orthopedic cases completed within 7 hours. *Anesthesiology* 103:401–405.

Van Slyke R, Wets R (1969) L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics* 17(4):638–663.

Vandenberghe M, Vuyst SD, Aghezzaf EH, Bruneel H (2019) Surgery sequencing to minimize the expected maximum waiting time of emergent patients. *European Journal of Operational Research* 275(3):971–982.

Varmazyar M, Akhavan-Tabatabaei R, Salmasi N, Modarres M (2020) Operating room scheduling problem under uncertainty: Application of continuous phase-type distributions. *IISE Transactions* 52(2):216–235.

Watson JP, Woodruff DL (2011) Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science* 8(4):355–370.

Yuan Y, Sen S (2009) Enhanced cut generation methods for decomposition-based branch and cut for two-stage stochastic mixed-integer programs. *INFORMS Journal on Computing* 21(3):480–487.

Zhang Z, Xie X (2015) Simulation-based optimization for surgery appointment scheduling of multiple operating rooms. *IIE Transactions* 47(9):998–1012.

Appendix

A Proof of Proposition 1

Letting $\mathbf{U} = \{u_{st} : s, t \in S\}$ denote the set of precedence relations from the first-stage of the two-stage SMIP model, suppose that σ^F is the set of induction room assignments based on the first available induction room rule in Proposition 1. Furthermore, let $\sigma^* \neq \sigma^F$ denote the set of assignments in an optimal solution to the model defined by (1)-(19). The claim in Proposition 1 is that σ^* can be converted to σ^F without any change in the objective value.

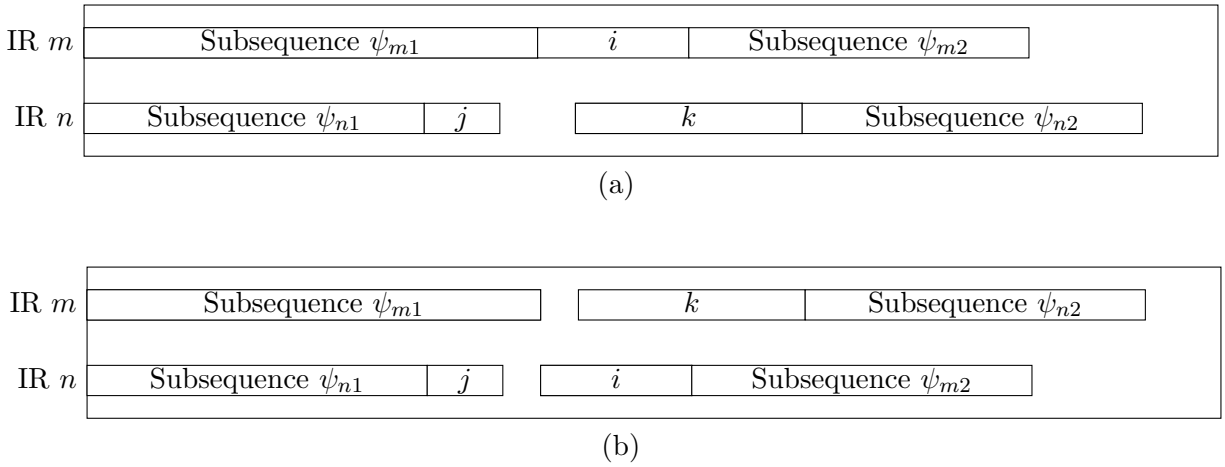

Figure A1: The patient assignments for induction rooms IR m and IR n (a) under σ^* and (b) after swapping

Since $\sigma^* \neq \sigma^F$, there has to be at least one surgery patient in σ^* for whom the induction room assignment will be different from that in σ^F. Let $i \in S$ be the earliest arriving such patient. Suppose that IR n is the first one that becomes available after (or upon) the arrival of $i \in S$. By construction, this patient is assigned to a different IR, say m, which becomes available later. Let $k \in S$ be the first patient assigned to IR n after the arrival of patient i, and patient $j \in S$ be the patient immediately preceding k on the set of assignments to IR n.

Figure A1 illustrates this situation. Let Subsequences ψ_{m1} and ψ_{m2} represent the set of assignments (which may include idle times) to IR m before and after the induction of patient i, respectively. Subsequences ψ_{n1} and ψ_{n2} denote the set of assignments to IR n before the induction of patient j and after that of patient k, respectively.

The induction of patient j must finish before the start of induction for patient i in IR m in Figure A1, as otherwise this would be a contradiction to IR n being the first available IR for patient i. Note further that by Constraints (8), we have $a_i < a_k$.

Consider the swapping of the assignments of patient i and Subsequence ψ_{m2} on IR m with those of patient k and Subsequence ψ_{n2} on IR n. In this case:

- Total IR idle time does not change, as the IR closure times have been swapped between these two IRs and total induction duration is the same.
- Total OR idle time is the same, as the OR schedules are unaffected (all patients leave their IR at the same times as in σ^*).
- Total patient waiting time is also identical, as the patients start their inductions at the same

times as in σ^*.

To convert σ^* to σ^F, one simply needs to find the earliest arriving patient whose IR assignment is different from σ^F, then find the first patient assigned to the IR that was first available for the former patient, and swap the subsequences as in Figure A1 without changing the objective value. The process continues with the next earliest arriving patient with a different assignment, and so on. In the worst case, the conversion to σ^F can be made in at most $|S|$ steps without any change in the objective. □