MULTI-PERIOD APPOINTMENT PLANNING AND SCHEDULING IN HEALTHCARE

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

UTKU TARIK BILGIÇ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

DECEMBER 2019

Approval of the thesis:

MULTI-PERIOD APPOINTMENT PLANNING AND SCHEDULING IN HEALTHCARE

submitted by **UTKU TARIK BILGIÇ** in partial fulfillment of the requirements for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** ______________

Prof. Dr. Yasemin Serin
Head of Department, **Industrial Engineering** ______________

Assist. Prof. Dr. Sakine Batun
Supervisor, **Industrial Engineering, METU** ______________

Examining Committee Members:

Prof. Dr. Ömer Kırca
Industrial Engineering, METU ______________

Assist. Prof. Dr. Sakine Batun
Industrial Engineering, METU ______________

Prof. Dr. Sinan Gürel
Industrial Engineering, METU ______________

Assoc. Prof. Dr. Seçil Savaşaneril Tüfekci
Industrial Engineering, METU ______________

Assist. Prof. Dr. Ceren Tuncer Şakar
Industrial Engineering, Hacettepe University ______________

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Utku Tarık Bilgiç

Signature :

ABSTRACT

MULTI-PERIOD APPOINTMENT PLANNING AND SCHEDULING IN HEALTHCARE

Bilgiç, Utku Tarık
M.S., Department of Industrial Engineering
Supervisor: Assist. Prof. Dr. Sakine Batun

December 2019, 97 pages

Appointment planning and scheduling (APS) plays a crucial role in patient service quality as well as utilization of valuable resources in healthcare. In this study, we considered the integrated problem of appointment planning and scheduling in an outpatient procedure center (OPC) over a planning horizon of multiple periods. We formulated the problem as a two-stage stochastic mixed-integer linear program (SMILP) with uncertainty in surgery durations. The first-stage problem consists of period assignment of surgeries, sequencing of surgeries in each period and appointment time for each surgery. In the second stage, surgery durations are realized and cost of patient waiting and idle time and overtime of operating room (OR) are calculated accordingly. We used symmetry breaking constraints in order to achieve computational efficiency and considered solution methods such as L-Shaped method, L-Shaped based branch-and-cut method and Benders' decomposition. We also considered several heuristic methods including simple sequencing rules, hierarchical planning, and genetic algorithm. We tested the performance of the proposed solution methods and estimated the value of the stochastic solution, the expected value of perfect information

and the value of integrated planning by conducting extensive numerical experiments.

Keywords: appointment planning, appointment scheduling, healthcare, stochastic programming, genetic algorithm

ÖZ

SAĞLIK ALANINDA ÇOK DÖNEMLİ RANDEVU PLANLAMA VE ÇİZELGELEME

Bilgiç, Utku Tarık
Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Sakine Batun

Aralık 2019 , 97 sayfa

Randevu planlaması ve çizelgelemesi, hastalara verilen hizmetin kalitesinde ve önemli kaynakların verimli kullanılabilmesinde çok büyük rol oynar. Bu çalışmada, bir ayakta tedavi polikliniğinde, birden çok dönem için planlama ve çizelgeleme kararlarını beraber ele alan, iki aşamalı stokastik programlama modeli kullanılmıştır. Model, ameliyat sürelerindeki belirsizlikleri göz önünde bulundurmaktadır. Birinci aşama problemi ameliyat randevularının dönemlere atanması, aynı dönemdeki ameliyatların kendi arasında sıralanması ve her bir ameliyat için ayrılan süre kararlarını içerir. İkinci aşama probleminde ise ameliyat süreleri belli olduktan sonra hastaların bekleme süreleri, ameliyathanenin atıl kaldığı ve fazla mesaide kullanıldığı süreler belirlenerek maliyetler hesaplanır. Bu modelde, daha iyi çözüm süreleri elde etmek amacıyla dönemler arası simetriyi kırmak için çeşitli kısıtlar kullanılmıştır. Problem çözümünde klasik L-shaped yöntemi, dal ve kesik tabanlı L-shaped yöntemi, Benders ayrıştırma yöntemi ve bu yöntemlerin varyasyonları kullanılmıştır. Bunların yanında, basit sıralama kuralları, hiyerarşik planlama ve genetik algoritma gibi çeşitli sezgisel yöntemler kullanılmıştır. Kapsamlı sayısal deneylerle, yukarıda bahsedilen

çözüm yöntemlerinin performanslarını test edilmiş ve elde edilen sonuçlar ile stokastik çözümün değeri, eksiksiz bilginin beklenen değeri ve birleşik planlamanın değeri hesaplanmıştır.

Anahtar Kelimeler: randevu planlama, randevu çizelgeleme, sağlık hizmetleri, stokastik programlama, genetik algoritma

To my beloved family and friends...

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my supervisor Assist. Prof. Dr. Sakine Batun for her valuable guidance and contributions throughout this study. I am deeply indebted to her for being patient with me and supportive of this extended journey.

I would like to express my sincere appreciation to my parents T. Tarık Bilgiç and Hülya Bilgiç, and my little sister, Ece Naz Bilgiç, for their endless love and unconditional support. Without them always being there for me, completing this thesis and achieving more would not be ever possible.

I would like to thank Dr. Bo Zeng for always believing in me and teaching me the importance of persistence and hard work.

I am also thankful to many other people who contributed to this study directly or indirectly. I would like to thank Doğancan Yaşar for inspiring me with being passionate about searching for scientific knowledge and his continuous encouragement. I am thankful to Gökçe Özkan, Utku Girit, and Sami Özarık, with whom I started to this academic journey together, for always being supportive and cheering me up. I am also very grateful to my friends: Tolga Açılmış, Alperen Yıldızalp, Büşra Karakaya, Şeyma Gürkan, Farid Taghiyev, and Burcu Maden, who made this journey of master's degree a happy one and bringing joy to my life.

Last but not least, I would like to thank Tuğçe Erbay, Dilruba Ekici, Can Ekici and Barış Ötüş for being my family in Pittsburgh and helped me survive during difficult times. Also, I am thankful to my cousin, Yağmur Kökbudak, who was always ready to help me when I was feeling blue.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

A-BDA	Automatic Benders' Decomposition Algorithm
ADP	Approximate Dynamic Programming
APS	Appointment Planning and Scheduling
APSP	Appointment Planning and Scheduling Problem
AS	Appointment Scheduling
ASC	Ambulatory Surgery Center
BDA	Benders' Decomposition Algorithm
B&C LM	L-shaped Method in a Branch-and-Cut Framework
CoefA	Ascending Coefficient of Variation
CoefD	Descending Coefficient of Variation
DA	Duration Assignment
DEP	Deterministic Equivalent Problem
EEV	Expected Value of Expected Value Problem
EF	Extensive Formulation
EV	Expected Value Problem
EVPI	Expected Value of Perfect Information
GA	Genetic Algorithm
HLP	Higher Level Problem
IP	Integer Program
LLP	Lower Level Problem
LM	L-shaped Method
LP	Linear Program
LPT	Longest Procedure Time First
MP	Master Problem

MIP	Mixed Integer Program
MVC	Mean Value Cuts
MDP	Markov Decision Process
OAS	Outpatient Appointment Scheduling
OECD	Organization for Economic Co-operation and Development
OPC	Outpatient Procedure Center
OR	Operating Room
PA	Period Assignment
RMP	Restricted Master Problem
RP	Recourse Problem
S	Sequencing
SAA	Sample Average Approximation
SLP	Stochastic Linear Program
SMILP	Stochastic Mixed Integer Linear Programming
SP	Subproblem
SP-TS	Subproblems Decomposed Based on Periods and Scenarios
SPT	Shortest Procedure Time First
WS	Wait-and-See Solution
VarA	Ascending Variance of Procedure Times
VarD	Descending Variance of Procedure Times
VSS	Value of the Stochastic Solution

CHAPTER 1

INTRODUCTION

Appointment planning and scheduling problem (APSP) is defined as deciding on date and order of upcoming tasks for a server over a period. In APSPs, the main purpose is providing high utilization of the server while completing these tasks on time, considering properties of each of them. APSP is a significant problem in many operational contexts such as ports, military airfields, legal services, manufacturing or healthcare environments. For example, servers can be cranes in a port, airfield operations teams or doctors; and tasks can be cargo ships, airplanes or patient appointments respectively ([1]).

Healthcare expenses is a major component of countries' budget. On average, 9% of gross domestic product of a country is dedicated to health expenditures among Organization for Economic Co-operation and Development (OECD) members. This percentage varies between 17.2% (in United States) and 2.8% (in Indonesia) ([2]). In addition to costs, for improving the quality of life of their people, it is necessary for a country to improve and effectively manage all its health facilities. For the last few decades, operations research has been used to develop solutions for many healthcare challenges. These challenges, which range from locating health facilities, radiation treatment planning, nurse scheduling to organ donation and transplant problems, are surveyed in [3].

Due to the developments in medical technology, different types of healthcare facilities such as outpatient procedure centers (OPCs), also known as ambulatory surgery centers (ASCs), has become more important. In United States, from 1996 to 2006, rate of hospital-based surgery centers remained almost the same while in OPCs this rate increased about 300% with estimated 14.9 million visits ([4]). Type of procedures performed at these facilities are mostly from wide range of minimal or non-invasive procedures, such as endoscopy, tonsillectomy and orthopedic surgeries. Duration of these elective (deferrable, non-urgent) procedures are shorter and they allow patients to recover at home. In addition to the costs, infection rates and complication rates are also lower in these centers, compared to hospital-based centers. Since these centers have separate resources and personnel and are used for elective procedures mostly, they have greater flexibility in management compared to similar hospital-based centers. However, this brings many other challenges for better management ([5]).

Visit of a patient to an OPC for a procedure can be summarized in three stages: (i) intake, (ii) procedure and (iii) recovery. Intake starts with the patient's check-in and preparation for the procedure. For the procedure, patient is taken to the operating room (OR) and then goes through the procedure. Then, the patient is brought to the recovery room and discharged ([6], [7]). Planning of surgeries, related staff (i.e., surgical teams composed of surgeons, anesthetists, nurses and surgical technicians), physical resources (operating, intake and recovery rooms) and equipment resources are critical for better utilization of an OPC and higher service quality.

ORs are among the most costly resources of medical facilities and its planning is essential for OPC's services ([8]). For this reason, appointment systems are vital for OPCs. Number of patients to be accepted for each day, their arrival sequences and arrival times, OR reserved time for their procedures are few of the questions to be answered while operating an OPC. After these are decided and schedules are perpared, patients are called and informed about their procedure and given the date and time. In the appointment scheduling (AS) literature, the term *planning* is used for determining the date and the sequence of the procedure of a patient. *Scheduling* is used for deciding on their appointment durations after planning decisions are made.

The main source of uncertainty in appointment systems in OPCs is duration uncertainties in each stage of patient's visit (intake, procedure and recovery). Even though procedures are pre-determined for each patient and all related information is available, durations cannot be known with certainty. However, from past data, distributions of these durations can be determined. Furthermore, due to the elective nature of these surgeries, patients may not show up for their surgeries or they may be late. While deciding on schedules, OPC management should indeed consider and plan according to these factors.

In an OR of an OPC, idle time may occur between two consecutive procedures or after all patients' surgeries are completed before the end of the session (i.e., the end of the regular working hours). These idle times are indicators of underutilization. Also, if all scheduled procedures could not be completed before the end of session, the surgical team would work overtime, which is costly for the management.

In this thesis, we studied the appointment planning and scheduling problem in an OPC for multiple periods under uncertain procedure durations of patients. In our model, main decisions are (i) assigning patients to periods, (ii) deciding their position in their assigned period (sequencing) and (iii) setting their appointment times (scheduling). We have constructed a two-stage stochastic mixed integer linear programming (SMILP) model for our problem. Our objective function is composed of multiple objectives, which are patient waiting times, and idle and overtime of OR. Expected weighted sum of the related costs are minimized over all scenarios. In the first stage, period and position assignments and procedure duration decisions are made for all patients simultaneously. In the second stage, waiting, idle time and overtime costs are calculated for each scenario. We solved this problem with different exact solution methods and then we analyzed these optimal solutions through simple sequencing rules. We developed a hierarchical decision making heuristic where, at the beginning, period assignment problem is solved by only considering overtime cost and then for final sequencing and time allocation decisions, problem is solved optimally, considering overall cost. Also, we developed a genetic algorithm for APSP. Patient-to-period assignments are represented in the chromosomes and simple sequencing rules are used for evaluating the quality of the solutions. By these heuristic methods, we are solving the problem in a drastically shorter amount of time and reach high quality solutions.

We conducted extensive numerical experiments to develop insights for the following questions:

1. What is the importance of considering uncertainty in surgery durations in APSP?

2. What is the benefit of considering period assignment, sequencing, and appointment scheduling decisions simultaneously (rather than hierarchically)?

3. How do the cost parameters in the objective function affect the structure of the optimal schedules?

The remainder of this thesis is structured as follows. In Chapter 2, literature review about APSP in OPCs and our contributions to literature by this research is presented in addition to the brief information about stochastic programming and genetic algorithm. In Chapter 3, detailed definition of the problem and formulation of the model are presented. Chapter 4 consists of explanations of the proposed solution methods in detail. In the Chapter 5, computational results and their analysis are given and discussed. In Chapter 6, we summarize our findings and briefly discuss the future research directions.

CHAPTER 2

LITERATURE REVIEW

2.1 Review of Studies on Appointment Planning and Scheduling

Appointment systems are widely used in both service and manufacturing systems for better management and higher utilization of significant resources. APSP is an important issue in wide variety of fields such as healthcare, production, logistics, project management and more. For example, in [9], for better traffic management and decreasing cargo waiting times, cargo ships are scheduled for a crane in a seaport. In another application in maritime, [10] suggests an appointment system for trucks to overcome marine terminal gate congestion, considering truck waiting times. In project management, [11] decides on earliest start time for each activity in a project, where durations of activities are uncertain. In a healthcare instance, [12] sets appointment times for patients of a chemotherapy clinic to minimize expected patient wait times while minimizing expected length of operations in a day.

In healthcare, appointments can be doctor visits or elective surgeries which can be planned in advance or non-elective surgeries/visits which are unexpected and should be taken care in short notice ([13]). Challenging factors of APSP and open research questions are reviewed in [13] in addition to the analysis of three most common appointment scheduling environments: (i) primary care, (ii) specialty care and (iii) elective surgery appointments. When the surgery scheduling is considered, another classification is based on the facility where patients will have these operations. In outpatient surgery setting, patients are admitted and discharged from the facility on the day of appointment surgery. These surgeries can be performed in ASCs in hospitals or OPCs. On the other hand, in inpatient surgeries, patients are hospitalized before surgery and recover in hospital for few more days ([14]). In comparison to inpatient setting, surgery start times and patient waitings before the procedures are more important than day of surgery for service quality and patient satisfaction in outpatient surgeries. Unpunctuality of patients or no-shows/cancellations should be considered and prepared against by the facilities. Detailed surveys about appointment scheduling in outpatient setting can be found in [15] and [16]. [15] gathers previous problem formulations and modeling considerations in outpatient setting. [16] presents a updated review by classifying the studies based on the level of decisions included, environmental factors, modeling approaches and solution methods. For various procedures, OPCs are preferred rather than hospital settings since they get higher service quality with lower risk of infection and complication rates. Also, they are less costly ([5]). Due to these cost and health benefits of the centers, starting with work of [17], appointment scheduling in outpatient setting has been getting extensive amount of attention.

In addition to scheduling patients, there are other factors in appointment scheduling in outpatient setting such as scheduling staff (nurses, surgical teams and surgeons), physical resources (related equipments with operations) and facilities (intake rooms, procedure rooms and recovery rooms). When the focus is mostly on these, related problems are referred to as surgery scheduling or operating room scheduling problems. Extensive surveys([14, 18, 8, 19]) can be reviewed for further details.

[16] is the most recent and an extensive review paper on outpatient appointment systems in healthcare. In the remaining parts of the chapter, we will use their categorization about outpatient appointment scheduling (OAS). In outpatient setting, decisions can be classified in three hierarchical levels: (i) strategic level, (ii) tactical level and (iii) operational level. Strategic level decisions are mainly design decisions which effect long term plans for the facility. These decisions include number of resources, access policy and walk-ins. In AS problems, strategic level decisions are mostly taken as input. In the tactical level, appointment intervals, capacity allocated to different surgery groups, priority of patient groups, scheduling window and more are considered. These decisions can be described as medium term planning decisions. In the shortest term, day-to-day plans about each individual patient, there are operational level decisions. Decision of appointment sequencing, deciding on appointment start

times, days, patient-to-server/resource allocation, acceptance of patients in sequential (online) scheduling are examples of operational decisions. In the literature, combinations of tactical and operational decisions are studied to represent real life conditions with given strategic decisions as input. These decisions can be made sequentially or simultaneously.

For better representation of the scheduling environment, some environmental factors should be considered. Uncertainty of procedure durations is one of the main sources of complexity of the problem. In the prior studies, to simplify the problem, all surgeries are assumed to be same kind, which means considering all surgery durations as independent and identically distributed. Another source of uncertainty is no-show of patients or cancellation of surgeries. For instance, in [20], heterogeneous patients through different no-show rates are considered. To deal with this issue, overbooking is a solution however it may decrease the patient satisfaction. [21] presents benefits of overbooking at different clinic sizes (i.e., number of patients) and different no-show rates and cost environments in OPCs. In addition to these, patient unpunctuality is another factor to be considered.

In the literature, due to uncertainties in the nature of OAS problem, stochastic programming and stochastic dynamic programming are used extensively to model these problems. In [12], a two-stage stochastic programming model is used for modeling appointments in a chemotherapy clinic. They determine appointment times in the first stage and chair of the patient, waiting times and patients' discharge times are decided for each scenario in second stage. [22] uses a multi-stage stochastic linear program (SLP) for sequential appointment scheduling where appointment requests from customers defines the stages. [23] used Markov Decision Process (MDP) appointment scheduling of chemotherapy patients considering the due dates and time windows. Approximate Dynamic Programming (ADP) is used to overcome the intractability of the problem. In [24], they used MDP for dynamic appointment scheduling decisions for a clinic where patient no-shows and cancellations are considered.

OAS is a complex problem and it is not easy to reach the optimal solution or near-optimal solutions with deterministic error bounds. For gaining insights, better understanding and analyzing the problem for better practice, heuristic methods are highly used. For example, [25] used a heuristic appointment policy derived considering the optimal solution structure for appointment scheduling of patients with pre-determined sequences. Metaheuristics are also commonly used with simulation as a candidate solution evaluation mechanism. In [7], a discrete event simulation model is constructed to review implementation of various easy-to-implement heuristics and then used in bi-criteria genetic algorithm (GA) to find Pareto optimal set of solutions for surgery planning and scheduling. [26] worked on multi-stage OR scheduling by using three simulation-based tabu search algorithms.

Our focus on literature will be on the combination of appointment planning and scheduling decisions, and dealing with uncertainties through stochastic programming related to our research. The patient list is pre-determined and all patients are scheduled simultaneously in an offline manner. Next, we will present the papers most relevant to our research in detail.

[27] formulated the appointment scheduling problem for a single server as a two-stage SLP. In the objective function, customer waiting time, server idle time and server overtime are considered. The sequence of customers to be scheduled are given and the appointment times (job allowances) are determined under service time uncertainty. For the solution, they designed a L-shaped algorithm with sequential bounding approach, by exploiting the decomposable structure of the problem to have computational advantage. They performed computational experiment for exploring insights through different cost coefficients and different i.i.d. service times.

In [28], effects of simultaneous decisions of both surgery sequencing and scheduling are investigated. They have extended the SLP model in [27] into a SMILP by including sequencing decision. A binary decision variable for immediate precedence of surgeries are used. By usage of this decision variable, it is possible to assign different waiting time and idle time cost coefficients for different sequences of patients. Since the problem is NP-hard, they could not solve the instances with more than three patients. They tried different heuristics and compared them in different cost parameter settings. They concluded that sequencing surgeries in an order of increasing variance works well.

Another model considering simultaneous sequencing and scheduling decisions is [29].

They constructed a new position-based model instead of modeling through precedence based decision variables in [28]. In this formulation, there is no need for subtour elimination constraints. To strengthen their model, they calculated tighter upper bounds on waiting time and idle time decision variables. Also, they used sample average approximation (SAA) as the solution approach and proved that the problem is NP-complete. As alternative solution methods, they developed three different Benders'-based algorithms which use a simplified master problem heuristic for solving the first-stage in iterative steps of Benders' decomposition.

[30] considered determining optimal number of patients to be scheduled simultaneously in addition to sequencing and scheduling decisions by extending [28]. Also, patient no-shows considered along with surgery duration uncertainty. For strengthening the model formulation, they calculated upper bounds on second-stage decision variables. For exploiting the model more, mean value cuts for upper bound in L-shaped algorithm are used similar to [31]. In addition to L-shaped method, hybrid multi-cut L-shaped and branch-and-bound with progressive hedging are used for exact solution. For larger instances, they came up with simple heuristics for sequencing by considering no-show rate and variance of patients and solving optimal for scheduling for a fixed sequence.

The most recent work about this problem is [32] where they proposed a new two-stage SMILP formulation and compared their model with [30] and [29] and showed performance improvements.

In elective surgery planning, block scheduling and open scheduling are two main systems. In block scheduling, blocks of OR time is assigned to surgeons and surgical groups for their surgeries. However, in open scheduling, all surgeons submit their surgeries up until schedule will be generated. Final schedule is decided by OR manager for the whole surgical listing ([13], [33]). In the literature, for the multi-period problems, mostly block scheduling problems exist however these studies consider resource allocation more rather than surgery schedules and start times.

In [34], in multiple ORs, surgery sequencing and scheduling is considered in an integrated manner. In their study, they constructed a three-stage SMILP model, where binary precedence-based decision variables are used to model the sequencing decisions. In the first stage of the problem, OR opening decisions and patient to OR assignments are made. After reveal of case cancellation information, sequence of patients in each OR and surgery starting times are decided. In the last stage, after real surgery durations are revealed, actual surgery start times, waiting and idling times, and costs are calculated. Their model also considers time windows for patients and soft/hard session length constraints for ORs.

Identical OR case is similar to the case of multi-period single OR planning and scheduling problem where all periods are identical. In [34], due to having OR opening decision with a fixed cost, they could use soft/hard session deadlines for the surgeries to be scheduled since opening a new OR is also possible. Compared to our position based model, they constructed their model by using binary decision variables for precedence of surgeries. To solve their model, they used Benders' decomposition method with feasibility cuts because of the structure of their model. In their computational experiments, uniform distribution for surgery durations are used and a three different duration-levels are used to represent historical real data from a hospital with low number of scenarios (largest solved is 400 scenarios). They provide sensitivity analysis results and managerial insights based on these small instances. Compared to [34], we are able to solve larger instances with higher number of scenarios and different types of surgeries, which we believe is a more realistic representation of an OPC.

In our work, we are planning and scheduling patients from a predetermined list of patients for multiple periods (i.e. scheduling inside time blocks of OR assigned to surgeons/surgical groups). In this thesis, we constructed a two-stage stochastic programming model for appointment sequencing and scheduling decisions simultaneously under uncertain patient surgery durations which can be extended to the case with no-show probabilities. Through tight upper bounds, symmetry breaking constraints and valid inequalities, exact solution can be reached in acceptable time for small instances. By using simple sequencing rules, we gained insights about each decision in the model and developed a genetic algorithm which enables us to solve much larger instances and reach near optimal solutions with less than 1% optimality gap.

2.2 Background Information on Stochastic Programming

2.2.1 Stochastic Programming

Stochastic programming is a method for considering uncertainties in mathematical models. Stochastic parameters are used in the model for representing these uncertainties. Each possible realization of these stochastic parameters are added into the model explicitly by scenarios with a probability. The purpose of stochastic programming models is to minimize/maximize the expected value of objective function over these scenarios.

A two-stage stochastic program is the simplest and most commonly used framework in stochastic programming. This framework is used where a decision should be made before all the uncertainty is revealed at a point of time. Full information reveal separates the problem into two-stages. In the first-stage problem, decisions are made not knowing the exact information. However, in the second-stage, for fixed decisions of first-stage for each scenario, new decisions are made in the case of all uncertainties revealed. The general two-stage stochastic programming model is given in [35] as follows:

$$
\begin{aligned}
\min z = & c^T x + \mathbb{E}_\xi \left[\min q(\omega)^T y(\omega) \right] \\
s.t. \quad & Ax = b \\
& T(\omega)x + W(\omega)y(\omega) = h(\omega) \\
& x \geq 0, y(\omega) \geq 0
\end{aligned}
\tag{2.2.1}
$$

where x denotes the first-stage variables and $y(\omega)$ denotes second-stage variables for each realization ω. The objective function is composed of the first-stage cost $c^T x$ and the expected cost of second-stage, $\mathbb{E}_\xi \left[\min q(\omega)^T y(\omega) \right]$, which depends on each ω, realizations of ξ. ξ represents the uncertainties in the second stage and ω is a realization of ξ. In the model, first-stage constraint $Ax = b$ has its all parameters certain. In the second stage, stochasticity is represented by technology matrix $T(\omega)$, recourse matrix $W(\omega)$ and right hand side vector $h(\omega)$ through realizations of ξ.

It is also possible to represent the stochastic program in its deterministic equivalent form. Deterministic equivalent problem (DEP) of (2.2.1) is stated in [35] as follows:

$$\begin{aligned} \min z = & c^T x + \mathscr{Q}(x) \\ s.t. \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{2.2.2}$$

where

$$\mathscr{Q}(x) = \mathbb{E}_{\xi}\left[Q(x, \xi(\omega))\right] \tag{2.2.3}$$

and, for a given realization w,

$$Q(x, \xi(\omega)) = \min_{y} \left\{ q(\omega)^T y \,|W(\omega)y = h(\omega) - T(\omega)x \,, y \geq 0 \right\} \tag{2.2.4}$$

In the above model, $\mathscr{Q}(x)$ represents the expected value of second-stage problem depending on first-stage decisions, x.

By representing ξ through finite set of scenarios $k \in K$ with their probabilities p_k, we can write the extensive form of the problem. Extensive form (EF) formulation is stated by [35] as follows:

$$\begin{aligned} & \min c^T x + \sum_{k=1}^{K} p_k q_k^T y_k \\ & s.t. \quad Ax = b \\ & \quad T_k x + W_k y = h_k \quad\quad k = 1, \ldots, K \\ & \quad x \geq 0, y_k \geq 0 \quad\quad\quad k = 1, \ldots, K \end{aligned} \tag{2.2.5}$$

2.2.2 Solution Methods

2.2.2.1 L-shaped Method

L-shaped Method (LM) is a decomposition method where the main idea is to approximate the expected objective function value of second-stage problem, $\mathscr{Q}(x)$. In this method, the problem is decomposed into a main problem, restricted master problem (RMP), and set of subproblems (SPs). First-stage decision variables and constraints form RMP and for fixed first-stage decisions and for each scenario, second-stage decision variables and constraints form SPs. The method works by solving RMP to optimality first, obtaining a feasible solution for all SPs, if not adding feasibility cuts and resolving. After a feasible solution is found, SPs are solved and optimality cuts are generated if necessary. This process continues iteratively until no more cuts are generated and objective function value of RMP converges.

In [35], steps of the algorithm are given as follows:

Step 0 Set $r = s = v = 0$.

Step 1 Set $v = v + 1$ and solve

$$\min z = c^T x + \theta \tag{2.2.6}$$

$$s.t. Ax = b \tag{2.2.7}$$

$$D_l x \geq d_l \qquad l = 1, \ldots, r \tag{2.2.8}$$

$$E_l x + \theta \geq e_l \qquad l = 1, \ldots, r \tag{2.2.9}$$

$$x \geq 0 \qquad \theta \in \Re \tag{2.2.10}$$

In this formulation (2.2.8) denotes feasibility cuts and (2.2.9) denotes optimality cuts. Let (x^v, θ^v) be an optimal solution. If no optimality cut is present, set θ^v equal to $-\infty$ and exclude from calculation of x^v

Step 2 Check whether x^v is feasible for all second-stage problems. If not, add at least one feasibility cut by solving the following model and return to *Step 1*. For each scenario k:

$$\begin{aligned} \min w' \quad &= e^T v^+ + e^T v^- \\ s.t. \quad & W_k y + I v^+ - I v^- = h_k - T_k x^v \\ & y \geq 0,\ v^+ \geq 0,\ v^- \geq 0 \end{aligned} \tag{2.2.11}$$

where $e^T = (1, \ldots, 1)$. Let σ^v is the associated dual variable of the model. Feasibility cut (2.2.8) is defined by the following:

$$\begin{aligned} D_{r+1} &= (\sigma^v)^T T_k \\ d_{r+1} &= (\sigma^v)^T h_k \end{aligned} \tag{2.2.12}$$

Else, go to *Step 3*

Step 3 For each scenario k, solve the following problem

$$\begin{aligned} \min w \quad &= q_k^T y \\ s.t. \qquad & W_k y = h_k - T_k x^v \\ & y \geq 0 \end{aligned} \tag{2.2.13}$$

Let π_k^v be the values of dual variables of the optimal solution of the above model. For optimality cut (2.2.9), define

$$\begin{aligned} E_{s+1} &= \sum_{k=1}^{K} p_k \cdot (\pi_k^v)^T T_k \\ e_{s+1} &= \sum_{k=1}^{K} p_k \cdot (\pi_k^v)^T h_k \end{aligned} \tag{2.2.14}$$

If $\theta^v \leq e_{s+1} - E_{s+1} x^v$, add (2.2.9) and return to *Step 1*.

Else *STOP*; current solution x^v is optimal.

2.2.2.2 L-shaped Method in a Branch-and-Cut Framework

In the classical LM, RMP is solved optimally at each iteration. This means, a new branch-and-bound tree is created and explored for solving the mixed integer RMP every single time. However, using L-shaped method in a branch-and-cut (B&C) framework changes the cut addition mechanism to the RMP. There is no difference in the generation of feasibility or optimality cuts in LM or model used. In B&C LM, cuts are added each time a new integer feasible solution is found in a single branch-and bound tree. By this way, the information gained from previous steps are conserved in one big tree instead of creating it every time. This helps to decrease the computational time when RMP is hard to solve.

2.2.2.3 Benders' Decomposition Algorithm

Benders' decomposition algorithm (BDA) is a general algorithm for dealing with large-scale problems by decomposing the problem into a master problem and (many) subproblems [36]. LM is a special case of Benders' decomposition where the subproblem is decomposed into further scenario subproblems.

2.2.3 Expected Value of Perfect Information and the Value of the Stochastic Solution

In stochastic programming, first-stage decisions should be made before uncertainty is revealed. Expected value of perfect information (EVPI) is a concept about maximum amount that a decision maker would pay for gaining this accurate future information beforehand [37]. Let ξ has only one particular scenario realization and the problem is as follows:

$$\begin{aligned} \min \quad & z(x,\xi) = c^T x + \min\left\{q^T y \mid Wy = h - Tx,\ y \geq 0\right\} \\ s.t. \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{2.2.15}$$

where $\bar{x}(\xi)$ is optimal solution for (2.2.15), considering a single scenario and $z(\bar{x}(\xi),\xi)$ is the objective function value of the optimal solution. Since we have many realizations of ξ represented by many scenarios, we could find the expected value solution which known as *wait-and-see* (WS) solution.

$$WS = \mathbb{E}_{\xi} z(\bar{x}(\xi),\xi) \tag{2.2.16}$$

To find the value of information, we compare it with recourse problem (RP) (2.2.1) which is as follows:

$$RP = \min_{x} \mathbb{E}_{\xi} z(x,\xi) \tag{2.2.17}$$

and EVPI is defined as

$$EVPI = RP - WS \tag{2.2.18}$$

Solving stochastic programs are computationally expensive. Value of the stochastic solution (VSS) is a measure to determine the worth of solving the stochastic program, instead of solving a deterministic model where stochastic parameters replaced with their expected values. Expected value problem (EV) considers a single scenario $\bar{\xi}$ where $\bar{\xi} = E(\xi)$ as follows:

$$EV = \min_{x} z(x,\bar{\xi}) \tag{2.2.19}$$

where the optimal solution of EV will be $\bar{x}(\bar{\xi})$. The expected result of using EV solution (EEV) is as follows:

$$EEV = \mathbb{E}_{\xi}(z(\bar{x}(\bar{\xi}), \xi))) \tag{2.2.20}$$

The difference between using the EV solution and solving our stochastic programming model optimally will give us the VSS.

$$VSS = EEV - RP \tag{2.2.21}$$

2.3 Background Information on Genetic Algorithm

Metaheuristics are "higher" level heuristic which can be used for a wide range of complex problems to obtain promising solutions. All metaheuristics are mostly inspired by nature, they include stochastic components and they need some problem specific parameter tuning. It is important for a metaheuristic to balance between diversification (exploration) and intensification (exploitation) while searching through the solution space ([38]).

Genetic Algorithm (GA) is a metaheuristic algorithm, inspired theory of evolution by natural selection by Charles Darwin, initially developed in [39]. GA has its roots in the idea of survival of the fittest among a population. In GA, solutions of the problem are represented through *chromosomes* which are encodings of different solutions in arrays, and each chromosome is constituted of *genes*. These chromosomes are individuals of a *population*. As the algorithm proceeds, some chromosomes are selected from the population and their genes are recombined, which is called *crossover*. After crossover, new solutions, *offsprings* are generated. With some probability, an offspring can go through *mutation*, where its genes are changed. Solution quality of each chromosome is evaluated by a *fitness function* and *fitness* values are calculated. In the population, through crossover, the chromosomes with high fitness values are spread through next generations and the whole population converges. The pseudo code for a generic GA is as follows:

Algorithm 1 Genetic Algorithm

Generate $population\_size$ many $chromosomes$ for creating the initial $population$

Compute $fitness$ of each $chromosome$

while stopping criteria is not satisfied **do**

  Choose two parent $chromosomes$

  Apply $crossover$ for reproduction

  Generate $offsprings$

  With probability p_m, $mutation$ happens

  Calculate $fitness$ of $offsprings$

  Form next generation

end

CHAPTER 3

PROBLEM DEFINITION AND MATHEMATICAL FORMULATION

3.1 Problem Definition

In this thesis, we considered the integrated problem of appointment planning and scheduling over a multi-period planning horizon. Our particular focus is on the scheduling of surgeries in an OR in an OPC. Three different levels of decisions, (i) surgery-to-period assignment, (ii) surgery sequence in a period, and (iii) allocated duration for each surgery, are determined simultaneously in this problem.

Procedure is a general term which contains surgical or non-surgical operations. However, we are focusing on planning and scheduling of an OR and focusing on surgical procedures, surgeries. Throughout this thesis, we use the terms "procedure", "patient" and "surgery" interchangeably when referring to the *job* to be processed on the *server*, which is the OR.

An OPC can be specific for one surgical department or there can be different departments, some of whom may use the same OR for their surgeries. Different time periods are assigned for completing these surgeries in a week and all surgeries should be completed in these periods. These time periods are typically days or half-days. Aim of our research is to successfully sequence and schedule a finite and pre-determined set of elective surgeries into available OR time periods for department(s). After all decisions are made, patients are called and informed for the time of their surgeries. The setting for the problem is an offline scheduling environment where all arrangements are made before the first period starts. Urgent patients are assumed to be handled separately by allocating some ORs or blocks of OR time which is common in practice. Therefore, in our problem, while making planning and scheduling decisions, urgent and walk-in patients are not considered and future adjustments are not allowed in the schedules, during the planning horizon.

Due to the scope of our work, we assume that ORs are the bottleneck resources and we do not consider the intake or recovery phases of surgeries and their required resources. We assume every patient will be ready on-time for their surgeries on their determined time. Surgical teams and nurses are assumed to be ready for each operation regardless of their schedules. OR setup times are considered to be included the procedure times, so a surgery can start immediately after the previous one ends. An OR will stay idle if the surgery of a patient ends before the appointment time of the next patient and no surgery can start before their appointment time. In a period, all assigned surgeries should be completed either during regular time or overtime.

For the planning horizon, patients to be scheduled are pre-determined. Type of their surgery and other necessary information is known about them. However, the duration of their surgeries are uncertain and imperfect information is available through probability distributions. Since the elective surgeries do not involve medical emergencies, there is no priority between patients. Also, neither starting time, nor period preference is allowed among the patients. However, our model can easily be extended for the case where surgeon or patient preferences are considered.

In this problem, there are two opposing parties to be satisfied. Since OR is scarce and a valuable resource of an OPC, it should be highly utilized. No idle time between the surgeries or overtime/undertime of the OR is desired by OPC management. On the other side, patient satisfaction is an important issue and it is not desired for them to wait for their surgeries after their planned time. Minimizing idle time of the OR is a conflicts with minimizing waiting time of the patients. Patient waiting time minimization requires longer surgery durations which will not lead any surgeries to exceed its allocated time. On the other hand, this will lead to idle time of OR between surgeries and overtime to complete all surgeries. Considering the importance of both objectives, the tradeoff should be balanced.

3.2 Mathematical Formulation

We have constructed a two-stage SMIP for multi-period appointment planning and scheduling problem in an OPC. In particular, we assign each surgery to a position in a period and then by allocating surgery durations, we determine their planned surgery start times, i.e., appointment times.

Objective function of the problem is composed of three pieces, (i) patient waiting time cost, (ii) OR idle time cost and (iii) OR overtime cost. If a surgery has to start after its appointment time (due to unfinished previous surgery), patient has to wait for his/her surgery and this leads to waiting time cost. If a surgery ends before appointment time for the next surgery, then the OR will be idle until the arrival of the next patient, which leads to idle time cost. Last but not least, if all the surgeries cannot be finished within the regular working hours in the period, they should be finished in overtime, which results in overtime cost. All these cost cost components are combined into a simple objective function by using proper cost coefficients (weights).

In the first stage of the problem, all the decisions about period assignment, sequencing and scheduling are made. In the second-stage problem, after all first-stage decisions are fixed, waiting time of each patient and, idle time and overtime of OR are calculated for each scenario of surgery durations. The objective function is completely composed of second stage variables since the first stage does not involve any cost. Our first-stage problem is a mixed integer program (MIP), whereas second-stage problem is an easy to solve linear program (LP). The extensive form includes both stages and is therefore a MIP.

In the remaining part of this section, we introduce our notation used in our model, then present extensive formulation of our two-stage SMILP and then explain the model.

Indices

i: Index for surgeries

k: Index for positions

t: Index for time periods

s: Index for scenarios

Sets

S: Set of scenarios

T: Set of time periods

N^r: Finite set of surgeries to be scheduled (i.e. $i = 1, \ldots, N$)

N^d: Set of dummy surgeries to be scheduled to mark the end of the period (i.e. $i = N+1, \ldots, N+T$)

N: Set of all surgeries $N = N^r \cup N^d$

K: Set of positions ($|K| = N+1$)

Parameters

cw: Per unit waiting time cost

cs: Per unit idle time cost

co: Per unit overtime cost of OR

cu: Per unit undertime cost of OR

MP_k^s: Sufficiently large constants for patient waiting time in position k in scenario s

MS: Sufficiently large constant for OR idle time

dt: Length of available regular time in a period (session length)

d_i^s: Actual duration of surgery i in scenario s

d_i^{min}: Shortest duration of surgery i among all scenarios

d_i^{max}: Longest duration of surgery i among all scenarios

p^s: Probability of scenario s

Decision Variables

First Stage Decision Variables

$$y_{ikt} = \begin{cases} 1, \text{if patient } i \text{ is assigned to position } k \text{ in period } t \\ 0, \text{otherwise} \end{cases}$$

x_{kt}: Allocated surgery duration for k^{th} patient in period t

Second Stage Decision Variables

p_{ikt}^s: Waiting time of patient i in position k in period t in scenario s

s_{ikt}^s: Idle time of OR before patient i in position k in period t in scenario s

o_t^s: OR overtime in period t in scenario s

u_t^s: OR undertime in period t in scenario s

Mathematical formulation

$$\textbf{Min} \quad \sum_{s \in S} p^s \left[\sum_{t \in T} \left(\sum_{k \in K_t} \sum_{i \in N^r} (cw \cdot p^s_{ikt} + cs \cdot s^s_{ikt}) + co \cdot o^s_t + cu \cdot u^s_t \right) \right] \quad (3.2.1)$$

subject to

$$\sum_{i \in N} y_{ikt} \leq 1 \qquad k \in K, t \in T \quad (3.2.2)$$

$$\sum_{t \in T} \sum_{k \in K} y_{ikt} = 1 \qquad i \in N \quad (3.2.3a)$$

$$\sum_{k \in K} y_{(N+t)kt} = 1 \qquad t \in T \quad (3.2.3b)$$

$$\sum_{k \in K} k \cdot y_{(N+t)kt} \geq \sum_{k \in K} k \cdot y_{ikt} + 1 \qquad i \in N^r, t \in T \quad (3.2.4)$$

$$\sum_{i \in N} y_{ikt} \geq \sum_{i \in N} y_{i(k+1)t} \qquad k \in K, t \in T \quad (3.2.5)$$

$$\sum_{i \in N} p^s_{ikt} - \sum_{i \in N} s^s_{ikt} = \sum_{i \in N^r} p^s_{i(k-1)t} + \sum_{i \in N^r} d^s_i \cdot y_{i(k-1)t} - x_{(k-1)t} \qquad k \in K, t \in T, s \in S \quad (3.2.6)$$

$$o^s_t - g^s_t = \sum_{i \in N^r} \sum_{k \in K} d^s_i \cdot y_{ikt} + \sum_{i \in N^r} \sum_{k \in k} S^s_{ikt} - dt \qquad t \in T, s \in S \quad (3.2.7)$$

$$p^s_{i1t} = 0 \qquad i \in N, t \in T, s \in S \quad (3.2.8a)$$

$$s^s_{i1t} = 0 \qquad i \in N, t \in T, s \in S \quad (3.2.8b)$$

$$p^s_{ikt} \leq MP^s_k \cdot y_{ikt} \qquad i \in N, k \in K, t \in T, s \in S \quad (3.2.8c)$$

$$s^s_{ikt} \leq MS \cdot y_{ikt} \qquad i \in N, k \in K, t \in T, s \in S \quad (3.2.8d)$$

$$y_{ikt} \in \{0, 1\} \qquad i \in N, k \in K, t \in T \quad (3.2.9)$$

$$x_{kt} \geq 0 \qquad k \in K, t \in T (3.2.10)$$

$$p^s_{ikt}, s^s_{ikt} \geq 0 \qquad i \in N, k \in K, t \in T, s \in S (3.2.11)$$

$$o^s_t, u^s_t \geq 0 \qquad t \in T, s \in S (3.2.12)$$

Objective function (3.2.1) is composed of the expected cost of waiting time, idle time, overtime and undertime of each period for the whole planning horizon over all scenarios.

Constraint (3.2.2) and (3.2.3a) are for properly assigning each patient to their positions. They ensure that each patient will be assigned to a single position and at most one patient can be assigned to any position respectively. Constraint (3.2.5) is responsible for placing all surgeries in period to consecutive positions and assigning earlier positions first.

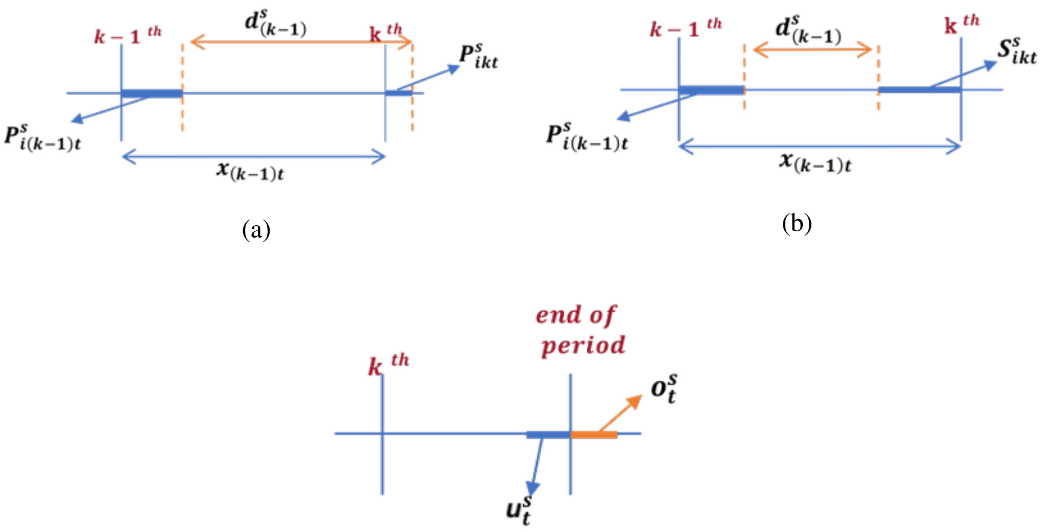
In our model, similar to [30], end-of-period dummy surgeries are defined. Actual length of a period is the sum of all surgery durations and idle times between surgeries for a period. However, after the last surgery, idle time should be excluded since the period ends when the last surgery is completed. Dummy surgeries are placed to end of each period by Constraint (3.2.3b) and Constraint (3.2.4) and they have specific surgery index $|N| + t$ for each period $t \in T$.

Constraints which are described above ((3.2.2), (3.2.3), (3.2.4), (3.2.5)) defines the first-stage problem. Second-stage problem is always feasible for the given assignments from the first stage. In the second stage problem, waiting time, idle time, overtime and undertime values are calculated for each scenario, for the given first-stage solution. In Figure 3.1, calculation of second-stage decision variables are illustrated on a timeline.

If a patient waits for another patient's surgery to end, patient waiting (P^s_{ikt}) occurs. Otherwise, if the surgery of previous patient ends earlier than the appointment time of the next patient, OR idle time (S^s_{ikt}) occurs (3.2.6). In any scenario s, for a position k on a period t, both waiting time and idle time will not take positive values since their columns are linearly dependent and both variables will lead an increase in objective function. In another way, we can define waiting and idle time for a position as follows:

$$p^s_k = \max\left(0, P^s_{k-1} + d^s_i \cdot y_{k-1} - x_{k-1}\right),$$
$$s^s_k = \max\left(0, x_{k-1} - P^s_k - d^s_i \cdot y_{k-1}\right).$$

The same discussion is valid for overtime and undertime, and both of these variables will not take positive value at the same time. In a period, if the surgeries cannot be



(a) (b) (c)

Figure 3.1: (a) Waiting time, (b) idle time, and (c) overtime, undertime calculations for a period.

completed within the regular times, then OR overtime is incurred. Conversely, if the surgeries are completed before the end of available time, the corresponding idle time of the OR considered as undertime (3.2.7). We can also define these variables as follows:

$$o_t^s = \max\left(0, \sum_{i=1}^{N}\sum_{k=1}^{N} d_{is}\cdot y_{ikt} + \sum_{i=1}^{N}\sum_{k=1}^{N} S_{ikts} - dt\right),$$

$$u_t^s = \max\left(0, dt - \sum_{i=1}^{N}\sum_{k=1}^{N} d_{is}\cdot y_{ikt} - \sum_{i=1}^{N}\sum_{k=1}^{N} S_{ikts}\right).$$

Since the first surgery of a period starts at time zero, there is no waiting and idle time for surgeries of first position as stated in Constraints (3.2.8a) and (3.2.8b). Constraint (3.2.8c) assures that, waiting time would be assigned to right patient and position on the right period and no empty positions would be assigned any waiting time. As a sufficiently large number for this constraint, we calculate an upper bound, MP_k^s, considering duration of surgeries in a scenario, minimum surgery duration and the position of the surgery.

$$MP_k^s = \left(\sum_{l=1}^{k-1}\left(dsc_l^s - d_l^{smin}\right)\right) \tag{3.2.13}$$

where

dsc_k^s: *k^{th} longest surgery duration in scenario s and*

d_k^{smin}: *k^{th} shortest surgery duration after sorting d_i^{min} in increasing order.*

Proof of this bound can be found in Appendix B. Similarly, as a sufficiently large number for the Constraint (3.2.8d), we used the bound calculated in [29] where

$$MS = \max_{i \in N^r} \left(\max_{s \in S} d_i^s - \min_{s \in S} d_i^s \right) \tag{3.2.14}$$

Last but not least, Constraints (3.2.9) (3.2.10) (3.2.11) and (3.2.12) are non-negativity and set constraints.

3.3 Symmetry Breaking Constraints and Valid Inequalities

3.3.1 Bounds on $\mathbf{x_{kt}}$

Let surgery i' be assigned to the first position on period t. Then, for all scenarios, allocating more time than $d_{i'}^{max}$ or less than $d_{i'}^{min}$ for x_{1t} will cause extra idle or waiting time cost. Then, let i'' be assigned to the second position, after i'. This time, considering their total time, it is never a good idea to assign more than $d_{i'}^{max} + d_{i''}^{max}$ or less than $d_{i'}^{min} + d_{i''}^{min}$ for $x_{1t} + x_{2t}$. Proceeding in this manner, we can come up with the following valid inequalities for allocated durations:

$$\sum_{l=1}^{k} \sum_{i=1}^{N} d_i^{min} \cdot y_{ilt} \leq \sum_{l=1}^{k} x_{lt} \qquad k \in K, t \in T \tag{3.3.1a}$$

$$\sum_{l=1}^{k} x_{lt} \leq \sum_{l=1}^{k} \sum_{i=1}^{N} d_i^{max} \cdot y_{ilt} \qquad k \in K, t \in T \tag{3.3.1b}$$

The above equations provide lower and upper bounds on the sum of allocated surgery durations up to position k on day t. By using Equation (3.3.1b), an upper bound for each x_{kt} can be obtained as follows:

$$x_{kt} \leq \sum_{i \in N^r} MX_{ik} \cdot y_{ikt} \qquad k \in K, t \in T \tag{3.3.2}$$

where MX_{kt} is defined as:

$$MX_{kt} = \begin{cases} d_i^{max}, & k = 1, t \in T \\ \sum_{l=1}^{k-1} \left(\rho_l + d_i^{max}\right), & k \geq 2, t \in T \end{cases} \tag{3.3.3}$$

where ρ_l denotes the l^{th} value when $(d_i^{max} - d_i^{min})$ for each $i \in N^r$ are sorted in decreasing order. Equation (3.3.2) has a larger (looser) upper bound than Equation (3.3.1b) since it is derived from Equation (3.3.1b). However, it is for each x_{kt} variable for $k \in K, t \in T$. By using Constraint (3.2.6) and Equations (3.3.1), a lower bound for each x_{kt} is derived as follows:

$$\sum_{l=1}^{k} \sum_{i=1}^{N} d_i^{min} \cdot y_{ilt} \leq x_{lt} \qquad k \in K, t \in T \tag{3.3.4}$$

Equation (3.3.4) has the same lower bound compared to Equation (3.3.1a) and it is a tighter bound since it is for each x_{kt}. Also, both of the bounds ensure that no duration is assigned to an empty position. Proofs of these upper and lower bounds can be found in Appendix A in detail.

3.3.2 Symmetry Breaking Constraints

All periods are identical in our problem. Due to its combinatorial nature, it leads many identical solutions just with different period indexing. In other words, there is complete symmetry with respect to periods. It is highly critical to eliminate these equivalent solutions to achieve computational efficiency ([30, 31, 40]).

To overcome this symmetry issue, we consider two different sets of symmetry breaking constraints. The main idea in the first set of constraints is restricting the possible period assignments for some surgeries. We fix the surgery with lowest surgery index to the first period, then restrict the surgery with second lowest surgery index to be assigned to first or second periods and so on. These constraints eliminate identical solutions by bounding the feasible region without eliminating any unique feasible solution. It is given as follows:

$$\sum_{t=1}^{i} \sum_{k \in K} y_{ikt} = 1 \qquad i = 1, \ldots, T-1 \tag{3.3.5}$$

Second set of symmetry breaking constraints ensures that every period has at least as many surgeries as the next period. In this idea, the first period will accommodate the highest number of surgeries, then the second period and so on. Also, there is no setup cost for assigning surgeries to a period, which leads to no incentive in having an empty period while overloading another period with many surgeries ($|N| > |T|$ in a practical instance). So, there should be at least one surgery in a period. The corresponding constraints are given as follows:

$$\sum_{k \in K} k \cdot y_{(N+t)kt} \geq \sum_{k \in K} k \cdot y_{(N+t+1)k(t+1)} \qquad t \in T - \{T-1\} \quad (3.3.6)$$

$$\sum_{k \in K} \sum_{i \in N^r} y_{ikt} \geq \sum_{k \in K} \sum_{i \in N^r} y_{ik(t+1)} \qquad t \in T - \{T-1\} \quad (3.3.7)$$

Only one of Constraints (3.3.6) and (3.3.7) or both of them can be used together. However, we cannot use both Equation (3.3.5) and Equations (3.3.6) and/or (3.3.7). They might be eliminating some part of solution space which are unique, not repetition of other solutions, and may include the optimal surgery-to-period assignments. As an example, first surgery does not have to be in the period accommodating highest number surgeries.

By considering the computational performance of both symmetry breaking constraint sets, as it is presented in Chapter 5, using only Equation (3.3.7) performs best. By keeping that symmetry breaking scheme in mind, we derived additional valid inequalities which may improve the computational performance.

3.3.3 Valid Inequalities Derived from Symmetry Breaking Constraints

We defined the available sets of positions for both real surgeries (K_t^r) and dummy surgeries (K_t^d) on each period, where $K_t = K_t^r \cup K_t^d$ by considering Equation (3.3.7). Also, pf_t and pl_t, the smallest and the largest possible total number of surgeries (including both real and dummy surgeries) can be assigned to period t are defined respectively. So $K_t^d = \{pf_t, \ldots, pl_t\}$. Since the first period would have the highest total number of surgeries among all periods. In first period, pf_1 is calculated by considering the case of assigning surgeries to periods evenly. For other periods $t \geq 2$, $pf_t = 2$ since there should be at least one real and one dummy surgery in each period ((3.3.8)). For calculating the largest possible number of total surgeries, for the

Table 3.1: Unique combination of number of patients in each period (including end-of-period dummy variables)

Period 1	Period 2	Period 3
9	2	2
8	3	2
7	4	2
7	3	3
6	5	2
6	4	3
5	5	3
5	4	4

first period, case of assigning only two surgeries (one real, one dummy) to each other periods and assigning remaining surgeries to the first period is considered. For $t \geq 2$ periods, pf_t is calculated in a similar manner. The case of assigning a two surgeries (one real, one dummy) to later periods than t and splitting the remaining number of surgeries equally between t and previous periods. It is defined in (3.3.9).

$$pf_t = \begin{cases} \lceil \frac{N}{T} + 1 \rceil, & t = 1 \\ 2, & t \geq 2 \end{cases} \tag{3.3.8}$$

$$pl_t = \begin{cases} |N| - |T| + 2, & t = 1 \\ \lfloor \frac{(N-T)-(T-t)}{t} + 1 \rfloor, & t \geq 2 \end{cases} \tag{3.3.9}$$

To explain this symmetry breaking scheme by an example, let us consider a toy problem with 3 periods and 10 patients (13 patients including end-of-period dummy variables). There are actually 8 different combinations for number of surgeries on each day, as explicitly given in Table 3.1. For these combinations, sets for available positions of surgeries will be as following:

$K_1^d : \{5, 6, 7, 8, 9\}$ $\quad K_1^r = \{1, 2, 3, \dots, 7, 8\}$
$K_2^d : \{2, 3, 4, 5\}$ $\quad K_2^r = \{1, 2, 3, 4\}$
$K_3^d : \{2, 3, 4\}$ $\quad K_3^r = \{1, 2, 3\}$

Regarding the above symmetry breaking idea, following constraints are added to the model.

$$\sum_{k \in K_t^d} y_{(N+t)kt} = 1 \qquad t \in T \quad (3.3.10a)$$

$$\sum_{t \in T} \sum_{k \in K_t^r} y_{ikt} = 1 \qquad i \in N^r (3.3.10b)$$

$$\sum_{k \notin K_t^d} y_{(N+t)kt} = 0 \qquad t \in T \quad (3.3.11a)$$

$$\sum_{t \in T} \sum_{i \in N^r} \sum_{k \notin K_t^r} y_{ikt} = 0 \qquad (3.3.11b)$$

$$\sum_{k \in K_t} \sum_{i \in N^r} y_{ikt} \geq pf_t - 1 \qquad t \in T \quad (3.3.12a)$$

$$\sum_{k \in K_t} \sum_{i \in N^r} y_{ikt} \leq pl_t - 1 \qquad t \in T \quad (3.3.12b)$$

$$\sum_{t \in T} \sum_{k \in K_t} k \cdot y_{(N+t)kt} = N + T \qquad (3.3.13)$$

Constraints (3.3.10) defines the possible positions for real surgeries and dummy surgeries in each period. Similarly, Constraints (3.3.11) prevents real and dummy surgeries to be assigned to positions other than the one, in K_t^r and K_t^d, respectively. Constraint (3.3.12), bounds the number of real surgeries in each period. Lastly, in Constraint (3.3.13), total number of surgeries in a period should be equal to the position index of dummy variables in that period.

To find the best combination of these valid inequalities and pick the best performing model, we performed extensive preliminary computational experiments. These experiments are explained in detail in Chapter 5.1.1

for computational performance, we did an extensive computational analysis among the models and chose the one. The preliminary experiments and its results for model selection are given in Chapter 5 with more explanation.

CHAPTER 4

SOLUTION METHODS

To find an exact solution to our two-stage SMILP, we solve the Extensive Formulation, use decomposition methods such as L-shaped method (LM) and Benders' decomposition algorithm (BDA). To analyze the problem, simple sequencing rules are tested for different decisions in the model, which provides us further information about optimal schedules. Considering those results, we solve our integrated problem of planning and scheduling in a hierarchical manner. Last but not least, we developed a Genetic Algorithm (GA) to find near-optimal solutions in short amount of time.

4.1 Exact Solution Methods

4.1.1 Solving The Extensive Formulation

Extensive Formulation (EF) which is represented by Constraints (3.2.1) - (3.2.12), (3.3.2), (3.3.4), (3.3.7), (3.3.10)-(3.3.11) and (3.3.13), is a large MIP model where all scenario variables are explicitly included in the model. All scenarios are equally likely to occur and $\frac{1}{|S|}$ is used for scenario probabilities. For small instances (up to 10 surgeries, 3 periods and up to 300 scenarios), the problem is tractable and solved by CPLEX 12.7.1. However for larger problem instances (in terms of surgeries, periods and scenarios), extensive form cannot be solved in a reasonable amount of time. Therefore, we use L-shaped method and its variants and Benders' decomposition algorithm, which allows larger problem instances to be solved by exploiting the structure of the formulation.

4.1.2 Decomposition Methods

4.1.2.1 L-shaped Method

In the classical L-shaped Method, the problem is decomposed into the first-stage and second-stage problems, where the second-stage problem is further decomposed into subproblems (SPs) for each scenario as explained in detail in Chapter 2. In our first stage problem, surgery-to-period, surgery-to-position and duration-to-surgery decisions are made. After all these assignments are made, waiting time, idle time and overtime values are calculated in the second-stage problem for each scenario. The objective function is composed of the costs of waiting time, idle time and overtime. The initial Master Problem (MP) and SPs are given below:

Initial MP: $\min z = 0$

$s.t.$

$$\sum_{i \in N} y_{ikt} \leq 1 \qquad k \in K, t \in T \ (Eq.3.2.2)$$

$$\sum_{t \in T} \sum_{k \in K} y_{ikt} = 1 \qquad i \in N \ (Eq.3.2.3a)$$

$$\sum_{k \in K} y_{(N+t)kt} = 1 \qquad t \in T \ (Eq.3.2.3b)$$

$$\sum_{k \in K} k \cdot y_{(N+t)kt} \geq \sum_{k \in K} k \cdot y_{ikt} + 1 \qquad i \in N^r, t \in T \ (Eq.3.2.4)$$

$$\sum_{i \in N} y_{ikt} \geq \sum_{i \in N} y_{i(k+1)t} \qquad k \in K, t \in T \ (Eq.3.2.5)$$

$$x_{kt} \leq \sum_{i \in N^r} MX_{ik} \cdot y_{ikt} \qquad k \in K, t \in T \ (Eq.3.3.2)$$

$$\sum_{l=1}^{k} \sum_{i=1}^{N} d_i^{min} \cdot y_{ilt} \leq x_{lt} \qquad k \in K, t \in T \ (Eq.3.3.4)$$

$$\sum_{k \in K} \sum_{i \in N^r} y_{ikt} \geq \sum_{k \in K} \sum_{i \in N^r} y_{ik(t+1)} \qquad t \in T - \{T-1\} \ (Eq.3.3.7)$$

$$\sum_{k\in K_t^d} y_{(N+t)kt} = 1 \qquad t \in T\ (Eq.3.3.10a)$$

$$\sum_{t\in T}\sum_{k\in K_t^r} y_{ikt} = 1 \qquad i \in N^r\ (Eq.3.3.10b)$$

$$\sum_{k\notin K_t^d} y_{(N+t)kt} = 0 \qquad t \in T\ (Eq.3.3.11a)$$

$$\sum_{t\in T}\sum_{i\in N^r}\sum_{k\notin K_t^r} y_{ikt} = 0 \qquad (Eq.3.3.11b)$$

$$\sum_{t\in T}\sum_{k\in K_t} k\cdot y_{(N+t)kt} = N+T \qquad (Eq.3.3.13)$$

$$y_{ikt} \in \{0,1\} \qquad i \in N, k\in K, t\in T\ (Eq.3.2.9)$$

$$x_{kt} \geq 0 \qquad k\in K, t\in T\ (Eq.3.2.10)$$

For each scenario s:

$$\text{SP: } \min \quad p^s \cdot \sum_{t\in T}\left(\sum_{k\in K_t}\sum_{i\in N^r}(cw\cdot p^s_{ikt} + cs\cdot s^s_{ikt}) + co\cdot o^s_t + cu\cdot u^s_t\right)$$

$$s.t.$$

$$\sum_{i\in N} p^s_{ikt} - \sum_{i\in N} s^s_{ikt} = \sum_{i\in N^r} p^s_{i(k-1)t} + \sum_{i\in N^r} d^s_i\cdot y_{i(k-1)t} - x_{(k-1)t} \qquad k\in K, t\in T$$

$$(Eq.3.2.6)$$

$$o^s_t - g^s_t = \sum_{i\in N^r}\sum_{k\in K} d^s_i\cdot y_{ikt} + \sum_{i\in N^r}\sum_{k\in k} S^s_{ikt} - dt \qquad t\in T$$

$$(Eq.3.2.7)$$

$$p^s_{i1t} = 0 \qquad i\in N, t\in T$$

$$(Eq.3.2.8a)$$

$$s^s_{i1t} = 0 \qquad i\in N, t\in T$$

$$(Eq.3.2.8b)$$

$$p^s_{ikt} \leq MP^s_k \cdot y_{ikt} \qquad i\in N, k\in K, t\in T$$

$$(Eq.3.2.8c)$$

$$s^s_{ikt} \leq MS \cdot y_{ikt} \qquad i \in N, k \in K, t \in T \qquad (Eq.3.2.8d)$$

$$p^s_{ikt}, s^s_{ikt} \geq 0 \qquad i \in N, k \in K, t \in T \qquad (Eq.3.2.11)$$

$$o^s_t, u^s_t \geq 0 \qquad t \in T \qquad (Eq.3.2.12)$$

In the initial solution of MP, any feasible assignment is acceptable since there is no contribution to the objective function from first-stage variables. After the first iteration, to consider the cost from SPs, θ variable is introduced as the approximate expected second-stage cost, and becomes the objective function of the Restricted Master Problem (RMP) for the next iterations. Through this variable, the information from SPs will be transferred to RMP through cuts, which will iteratively lead to optimal solution as defined in Section 2.2.2.1. In each iteration, RMP is resolved after an optimality cut is generated from dual variables of the second-stage problem.

In Figure 4.1, flowchart of the iterative LM is given for our problem. In this problem, RMP is a MIP and SPs are linear programs. This enables obtaining optimality cuts from the dual solutions of SPs. For any feasible solution of first stage, second-stage subproblems are always feasible in our problem. Thus, our problem has relatively complete recourse, as defined in [35] and feasibility cuts are not required to be generated. This brings a computational advantage in the method. SPs are easy to solve linear models and waiting and idle time and overtime can be calculated without solving LPs. However, for the optimality cut generation, dual variables are required and LP should be solved for each SP.

Instead of solving RMP from scratch in every iteration, L-shaped method can be implemented in a Branch-and-Cut (B&C) framework. In B&C framework, only one branch-and-bound tree is generated and explored. Cuts are added at every incumbent solution and stored to be used while exploring the other nodes. Flowchart of the L-shaped Method in B&C framework (B&C LM) is given in Figure 4.2.

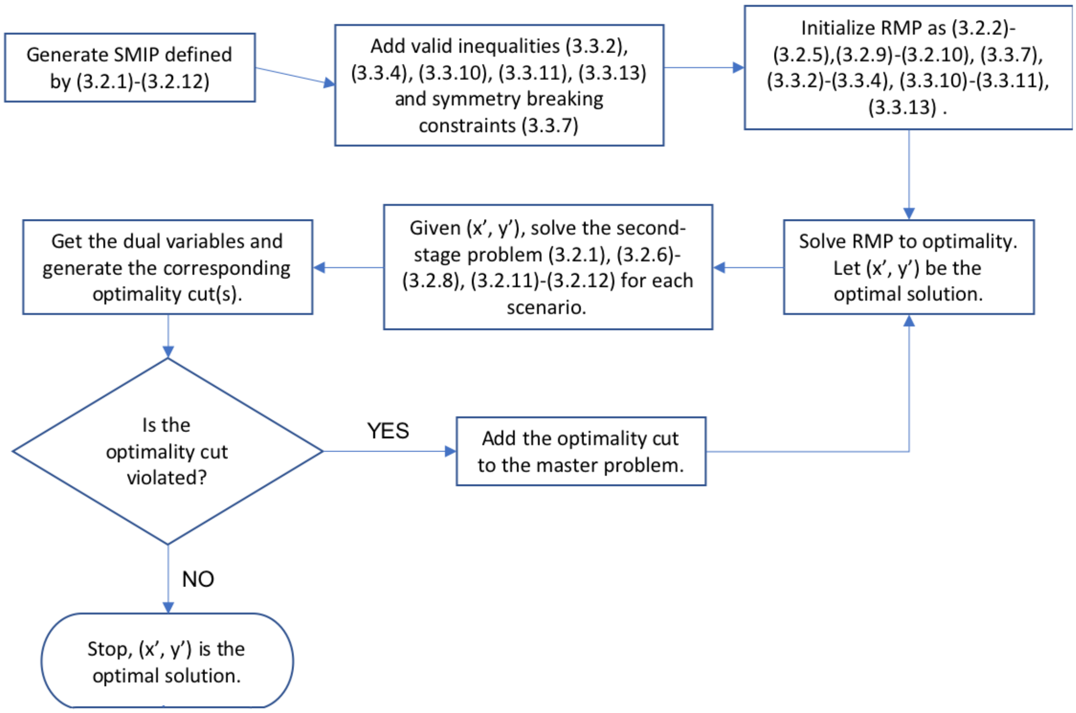


Figure 4.1: Flowchart representation of the iterative L-shaped algorithm

4.1.2.2 Decomposition Schemes for the Subproblems

In our problem, once the first-stage decisions are made, each period is independent from each other. After first-stage decisions are fixed, SPs can be decomposed further into easier to solve smaller subproblems for each period t. After planning and scheduling decisions are made, waiting time, idling time and overtime for each period can be calculated independently for every scenario. The further decomposed SPs (SP-TS), considering both scenario and period decomposition are given below. For each scenario s and each period t:

$$\text{SP-TS: } \min \frac{1}{|S|} \cdot \sum_{k \in K_t} \sum_{i \in N^r} (cw \cdot p^s_{ikt} + cs \cdot s^s_{ikt}) + cg \cdot o^s_t + cu \cdot u^s_t$$

$$s.t.$$

$$\sum_{i \in N} p^s_{ikt} - \sum_{i \in N} s^s_{ikt} = \sum_{i \in N^r} p^s_{i(k-1)t} + \sum_{i \in N^r} d^s_i \cdot y_{i(k-1)t} - x_{(k-1)t} \qquad k \in K \; (Eq.3.2.6)$$

$$o^s_t - g^s_t = \sum_{i \in N^r} \sum_{k \in K} d^s_i \cdot y_{ikt} + \sum_{i \in N^r} \sum_{k \in k} S^s_{ikt} - dt \qquad (Eq.3.2.7)$$

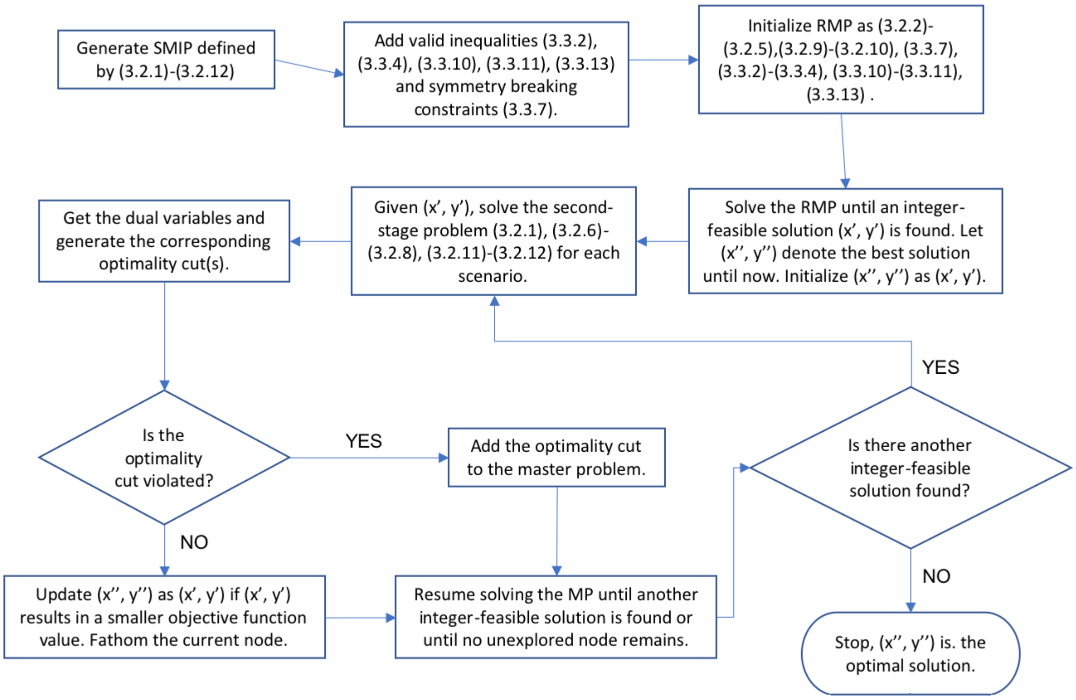


Figure 4.2: Flowchart representation of the L-shaped algorithm in the branch-and-cut framework

$$p^s_{i1t} = 0 \qquad i \in N \ (Eq.3.2.8a)$$

$$s^s_{i1t} = 0 \qquad i \in N \ (Eq.3.2.8b)$$

$$p^s_{ikt} \leq MP^s_k \cdot y_{ikt} \qquad i \in N, k \in K \ (Eq.3.2.8c)$$

$$s^s_{ikt} \leq MS \cdot y_{ikt} \qquad i \in N, k \in K \ (Eq.3.2.8d)$$

$$p^s_{ikt}, s^s_{ikt} \geq 0 \qquad i \in N, k \in K \ (Eq.3.2.11)$$

$$o^s_t, u^s_t \geq 0 \qquad (Eq.3.2.12)$$

4.1.2.3 Multi-Cut Structure

In LM, after each iteration, a single optimality cut (2.2.9) is generated and added to the RMP. By this way, the size of RMP increases one constraint at a time. However, the information transferred through one cut is limited and this leads to high number of iterations. In the multi-cut approach, more information can be transferred through multiple cuts (i.e., a cut for each scenario, period, or scenario-period combination) from SP to RMP. This will decrease the number of iterations in the method. However the RMP will be more difficult to solve compared to the single-cut method.

In the multi-cut L-shaped method (MLM), instead of a single cut, a cut is added for each scenario using θ_s $\left(\sum_{s\in S}\theta_s \geq \dots\right)$ and the RMP objective function becomes $\sum_{s\in S}\theta_s$. In multi-cut SP-TS decomposition (MLM-TS), there will be $|T| \times |S|$ cuts generated in each iteration $\left(\sum_{s\in S}\theta_{ts} \geq \dots \quad t \in T\right)$. This time RMP objective function becomes $\sum_{t\in T}\sum_{s\in S}\theta_{ts}$.

4.1.2.4 Benders' Decomposition Algorithm

In LM, planning and scheduling decisions are made by y_{ikt} and x_{kt} variables for the whole planning horizon, before actual surgery durations are revealed. However, to achieve computational efficiency, the problem could be decomposed in a way that only y_{ikt} variables can be kept in the MP and x_{kt} variables can be handled in the subproblems. This decomposition structure is used in Automatic Benders' Decomposition Algorithm embedded in CPLEX. First-stage problem is a pure integer program (IP). Since x_{kt} variables are scenario independent, further scenario decomposition is not possible. Accordingly, Benders' Decomposition Algorithm (BDA) can be implemented by using period-based SPs as follows:

$$\min \quad \sum_{t\in T} q_t(p, s, o, u)$$

$$s.t.$$

$$\sum_{i\in N} y_{ikt} \leq 1 \qquad k \in K, t \in T \ (Eq.3.2.2)$$

$$\sum_{t\in T}\sum_{k\in K} y_{ikt} = 1 \qquad i \in N \ (Eq.3.2.3a)$$

$$\sum_{k\in K} y_{(N+t)kt} = 1 \qquad t \in T \ (Eq.3.2.3b)$$

$$\sum_{k\in K} k \cdot y_{(N+t)kt} \geq \sum_{k\in K} k \cdot y_{ikt} + 1 \qquad i \in N^r, t \in T \ (Eq.3.2.4)$$

$$\sum_{i\in N} y_{ikt} \geq \sum_{i\in N} y_{i(k+1)t} \qquad k \in K, t \in T \ (Eq.3.2.5)$$

$$\sum_{k\in K}\sum_{i\in N^r} y_{ikt} \geq \sum_{k\in K}\sum_{i\in N^r} y_{ik(t+1)} \qquad t \in T - \{T-1\} \ (Eq.3.3.7)$$

$$\sum_{k \in K_t^d} y_{(N+t)kt} = 1 \qquad t \in T \ (Eq.3.3.10a)$$

$$\sum_{t \in T} \sum_{k \in K_t^r} y_{ikt} = 1 \qquad i \in N^r \ (Eq.3.3.10b)$$

$$\sum_{k \notin K_t^d} y_{(N+t)kt} = 0 \qquad t \in T \ (Eq.3.3.11a)$$

$$\sum_{t \in T} \sum_{i \in N^r} \sum_{k \notin K_t^r} y_{ikt} = 0 \qquad (Eq.3.3.11b)$$

$$\sum_{t \in T} \sum_{k \in K_t} k \cdot y_{(N+t)kt} = N + T \qquad (Eq.3.3.13)$$

$$y_{ikt} \in \{0, 1\} \qquad i \in N, k \in K, t \in T \ (Eq.3.2.9)$$

where

$$q_t(p, s, o, u) = \min \sum_{s \in S} p^s \left(\sum_{k \in K_t} \sum_{i \in N^r} (cw \cdot p_{ikt}^s + cs \cdot s_{ikt}^s) + cg \cdot o_t^s + cu \cdot u_t^s \right)$$

$s.t.$

$$x_{kt} \leq \sum_{i \in N^r} MX_{ik} \cdot y_{ikt} \qquad k \in K, t \in T$$

$$(Eq.3.3.2)$$

$$\sum_{l=1}^{k} \sum_{i=1}^{N} d_i^{min} \cdot y_{ilt} \leq x_{lt} \qquad k \in K, t \in T$$

$$(Eq.3.3.4)$$

$$x_{kt} \geq 0 \qquad k \in K, t \in T$$

$$(Eq.3.2.10)$$

$$\sum_{i \in N} p_{ikt}^s - \sum_{i \in N} s_{ikt}^s = \sum_{i \in N^r} p_{i(k-1)t}^s + \sum_{i \in N^r} d_i^s \cdot y_{i(k-1)t} - x_{(k-1)t} \qquad k \in K, t \in T$$

$$(Eq.3.2.6)$$

$$o_t^s - g_t^s = \sum_{i \in N^r} \sum_{k \in K} d_i^s \cdot y_{ikt} + \sum_{i \in N^r} \sum_{k \in k} S_{ikt}^s - dt \qquad t \in T$$

$$(Eq.3.2.7)$$

$$p_{i1t}^s = 0 \qquad i \in N, t \in T$$

$$(Eq.3.2.8a)$$

$$s_{i1t}^s = 0 \qquad i \in N, t \in T$$

$$(Eq.3.2.8b)$$

$$p_{ikt}^s \leq MP_k^s \cdot y_{ikt} \qquad i \in N, k \in K, t \in T$$

$$(Eq.3.2.8c)$$

$$s_{ikt}^s \leq MS \cdot y_{ikt} \qquad i \in N, k \in K, t \in T$$

$$(Eq.3.2.8d)$$

$$p_{ikt}^s, s_{ikt}^s \geq 0 \qquad i \in N, k \in K, t \in T$$

$$(Eq.3.2.11)$$

$$o_t^s, u_t^s \geq 0 \qquad t \in T \ (Eq.3.2.12)$$

Similar to LM, for any feasible first-stage decisions, second stage is always feasible and hence no feasibility cuts are needed in BDA as well. For each surgery-to-period assignment, all duration allocation decisions are feasible. Here, the first-stage problem becomes pure IP instead of MIP and the second stage is still LP with extra variables and constraints. In BDA, SPs are more complex since time allocation decisions are added. However, RMP has less variables and constraints. Solution time for the first stage will decrease, whereas SP solution times will increase.

4.1.2.5 Strengthening the Master Problem using Mean Value Cuts

In [31], for speeding up the convergence of LM, valid inequalities based on mean value scenario $\bar{\xi}$ are proposed. These inequalities are inspired from Jensen's Inequality, which provides a lower bound for expected second-stage cost. For better computational performance, we added these cuts to our LM and its variants. With the addition of these cuts, first-stage problem returns better solutions in terms of expected second-stage performance.

For the implementation of mean value cuts (MVC), additional decision variables and parameters are required. These decision variable are mean value scenario specific versions of second-stage decision variables. Also, lower and upper bound parameters are modified for $\bar{\xi}$. All these variables and inequalities, which are presented below, are added to the RMP.

Parameters:

- $\bar{d}_i$: Expected duration of surgery i
- $\bar{MP}_k$: Sufficiently large constants for patient waiting time under the mean value scenario in position k
- $\bar{MS}$: Sufficiently large constant for OR idle time under mean value scenario

Decision Variables:

- $\bar{p}_{ikt}$: Waiting time of patient i in position k in period t under the mean value scenario
- $\bar{s}_{ikt}$: Idle time of OR before patient i in position k in period t under the mean value scenario
- $\bar{o}_t$: OR overtime in period t under the mean value scenario
- $\bar{g}_t$: OR undertime in period t under the mean value scenario

Inequalities:

$$\sum_{i\in N}\bar{p}_{ikt} - \sum_{i\in N}\bar{s}_{ikt} = \sum_{i\in N^r}\bar{p}_{i(k-1)t} + \sum_{i\in N^r}\bar{d}_i \cdot y_{i(k-1)t} - x_{(k-1)t} \qquad k \in K_t, t \in T$$

$$\bar{o}_{ts} - \bar{g}_{ts} = \sum_{i=1}^{N}\sum_{k=1}^{N} d_{is}\cdot y_{ikt} + \sum_{i=1}^{N}\sum_{k=1}^{N}\bar{s}_{ikt} - dt \qquad t \in T$$

$$\bar{p}_{i1t} = 0 \qquad i \in N, t \in T$$

$$\bar{s}_{i1t} = 0 \qquad i \in N, t \in T$$

$$\bar{p}_{ikt} \leq \bar{MP}_k \cdot y_{ikt} \qquad i \in N, k \in K_t, t \in T$$

$$\bar{s}_{ikt} \leq \bar{MS}\cdot y_{ikt} \qquad i \in N, k \in K_t, t \in T$$

$$\bar{p}_{ikt}, \bar{s}_{ikt} \geq 0 \qquad i \in N, k \in K_t, t \in T$$

$$\bar{o}_t, \bar{u}_t \geq 0 \qquad t \in T$$

Then,

$$\theta \geq \sum_{t \in T} \left(\sum_{k \in K_t} \sum_{i \in N^r} (cw \cdot \bar{p}_{ikt} + cs \cdot \bar{s}_{ikt}) + co \cdot \bar{o}_t + cu \cdot \bar{u}_t \right) \tag{4.1.1}$$

In SP-TS, valid inequality (4.1.1) is modified as follows for each $t \in T$:

$$\sum_{s \in S} \theta_{ts} \geq \sum_{k \in K_t} \sum_{i \in N^r} (cw \cdot \bar{p}_{ikt} + cs \cdot \bar{s}_{ikt}) + co \cdot \bar{o}_t + cu \cdot \bar{u}_t \tag{4.1.2}$$

4.2 Heuristic Solution Methods

Even though decomposition methods provide a good alternative for dealing with the problem, computational time and memory requirements of these methods can still be huge. For finding well performing solutions in a reasonable amount of computational time, we considered heuristic solution methods.

4.2.1 Simple Sequencing Rules

We tried six different sequencing rules which are also used in [7]. These rules are:

- SPT: Shortest procedure time first
- LPT: Longest procedure time first
- VarA: Ascending variance of procedure times
- VarD: Descending variance of procedure times
- CoefA: Ascending coefficient of variation
- CoefD: Descending coefficient of variation

where coefficient of variation is the ratio of standard deviation to mean ($c^v = \frac{\sigma}{\mu}$). These sequencing rules are used for planning (assignment, sequencing) decisions in the problem.

For surgery duration allocation, similar to [7], we have considered 3 different percentiles ($25^{th}(p25)$, $50^{th}(p50)$ and $75^{th}(p75)$) and mean value (μ) of procedure times.

This is called *job hedging* and it has been widely used in single server scheduling literature.

To explain our work clearly, we developed a representation for our first stage decisions in a sequential manner. It includes all planning and scheduling decisions: (i) Period Assignment (PA), (ii) Sequencing (S) and (iii) Duration Assignment (DA). Our representation is $PA/S/DA$. PA decision is similar to bin packing problem in which each period is a *bin* and surgeries are *items* to be placed. When making the assignments, it is required to determine the *weight* of each *item* to determine the bin with the nearest availability. In S, for each period, patients are sequenced based on the specified sequencing rule. Last but not least, DA indicates how much time should be allocated for a surgery.

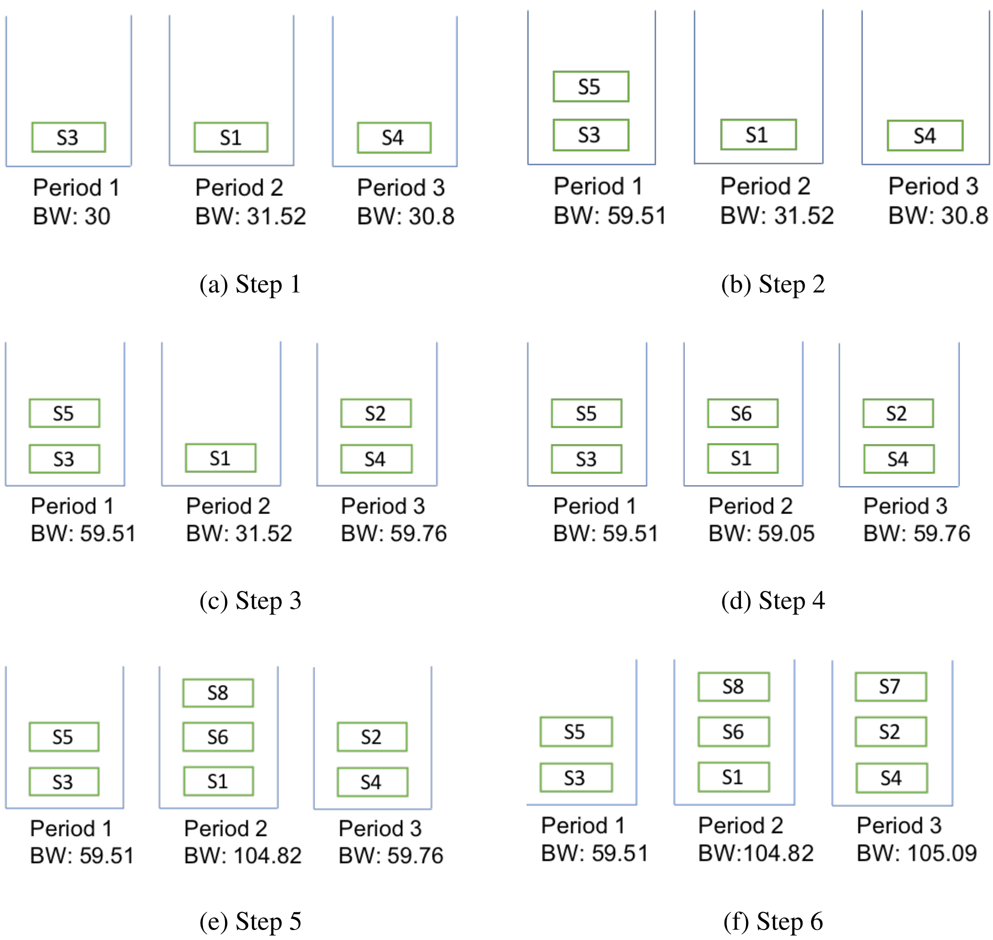
For better clarification of the representation, we explain it through an example. Let us consider *"VarA-p25/SPT/μ"* for planning and scheduling 8 surgeries into 3 periods. The list of surgeries, together with their mean, standard deviation, and 25^{th} percentile values, are given in Table 4.1. First, surgeries in the surgery list are sorted in ascending order, considering the variance of their durations (VarA) (Table 4.2). Then starting from the first surgery, one surgery is assigned to the period with nearest availability. When doing so, the sum of 25^{th} percentile values of the surgery duration in each period are compared and the decision is made accordingly as it is given in Figure 4.3. After all surgery-to-period assignments are made, surgeries are sequenced considering their durations in each period based on the SPT rule. Similar to surgery durations, 25^{th} percentile is considered here as well. At this level, y_{ikt} variables are assigned in the model. For the duration, the mean procedure duration of each assigned surgery will be assigned as indicated by μ in the representation. The final solution is given in Figure 4.4. After all the first-stage decisions are made, it is straightforward to calculate waiting, idle times and overtime and then their costs.

Table 4.1: Surgeries to be scheduled into 3 periods

Surgery	μ	σ	$p25$
S1	42.13	15.33	31.52
S2	41.01	17.10	28.96
S3	40.05	15.26	30.00
S4	42.75	16.24	30.80
S5	41.55	17.09	29.50
S6	41.50	18.27	27.53
S7	76.71	43.58	45.33
S8	74.00	41.03	45.76

Table 4.2: Sorted surgical list with ascending variance (VarA) rule

Surgery	μ	σ	$p25$
S3	40.05	15.26	30.00
S1	42.13	15.33	31.52
S4	42.75	16.24	30.80
S5	41.55	17.09	29.50
S2	41.01	17.10	28.96
S6	41.50	18.27	27.53
S8	74.00	41.03	45.76
S7	76.71	43.58	45.33



(a) Step 1 (b) Step 2

(c) Step 3 (d) Step 4

(e) Step 5 (f) Step 6

Figure 4.3: Example of Period Assignment decision

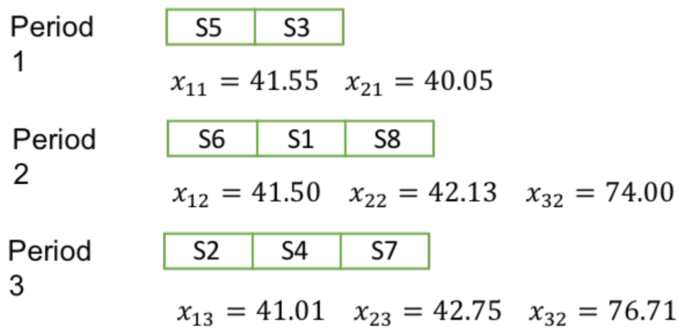


Figure 4.4: Final sequence of each surgery in each period and allocated surgery durations

4.2.2 Hierarchical Decision Making

In the APSP, we consider integrated decisions of planning and scheduling. We developed a heuristic algorithm where we make decisions hierarchically in two phases. This will also help us understand the effect of integration in APSP. The decisions in the problem are grouped into two problems. Higher Level Problem (HLP) includes only the determination of periods of the surgeries with the objective of minimizing expected overtime cost. In the Lower Level Problem (LLP), with the given period assignments returned by HLP, sequencing and scheduling decisions are made to minimize the expected server idling and patient waiting cost.

For the HLP, we solve the model as given below:

Decision Variables

$$w_{it} = \begin{cases} 1, \text{if patient } i \text{ is assigned to period } t \\ 0, \text{otherwise} \end{cases}$$

ot_t^s: Overtime of period t in scenario s

Parameters

co: Per unit overtime cost of OR

dt: Length of available regular time in a period (session length)

d_i^s: Actual duration of surgery i in scenario s

$$
\begin{aligned}
&Min \sum_{s\in S} p^s \cdot \sum_{t\in T} co \cdot o_t^s && \\
&\quad s.t. \sum_{t\in T} w_{it} = 1 && i \in N^r \\
&\qquad ot_t^s \geq \left(\sum_{i\in N^r} d_i^s \cdot w_{it}\right) - dt && t \in T, s \in S \\
&\qquad w_{it} \in \{0,1\} && i \in N^r, t \in T \\
&\qquad ot_t^s \geq 0 && t \in T, s \in S
\end{aligned}
$$

with symmetry breaking constraints

$$
\sum_{i\in N^r} w_{it} \geq \sum_{i\in N^r} w_{i(t+1)} \qquad t = 1,\ldots,T-1
$$

After getting the period assignment information (w_{it}) from HLP, we solve LLP, which is APSP where the period assignment is fixed using the following constraint:

$$
\sum_{k\in K} y_{ikt} = w_{it} \qquad i \in N^r, t \in T \quad (4.2.1)
$$

Then we solve LLP by using BDA to improve computational efficiency. We also create a random sequence from the given period assignments and feed this solution as an advanced starting point for the optimization of LLP, which accelerates computational performance.

4.2.3 Genetic Algorithm

In light of analysis regarding simple sequencing rules and hierarchic decision making, we developed a Genetic Algorithm (GA) for APSP for solving larger instances (in the sense of higher number of scenarios and/or higher number of periods and/or surgeries) in reasonable times and finding near-optimal solutions. Our GA searches the solution space and try to reach near-optimal surgery-to-period assignments. After finding a "good" surgery-to-period assignment, sequencing and appointment scheduling decisions are solved optimally for each period separately.

Representation Scheme

The representation has the idea of a bin packing problem where each period is a bin and surgeries are the items. There are $|N|$ patients and we are using a $1 \times |N|$ array where each gene represents the period of each surgery. They can take values between 1 to $|T|$. An example for 8 surgeries to be assigned to 3 periods are given in Figure 4.5.

Surgery	1	2	3	4	5	6	7	8
Period	1	2	2	3	1	3	2	3

Figure 4.5: Example chromosome representation

In this example, surgeries with index 1 and 5 are in the first period, $2,\ 3$ and 7 are in the second period and $4,\ 6$ and 8 are in the last period. This chromosome representation does not include any information about the sequence or appointment time of surgeries in any period.

Fitness Function

To evaluate the solution quality of each solution, we use the same expected weighted sum of waiting, idling, and overtime costs as the fitness function in Equation (3.2.1). Higher fitness values correspond to lower quality solutions. In order to calculate these costs, sequence of the surgeries in each period and their allocated surgery times are required. It is possible to calculate the optimal objective function value where surgery-to-period assignments are fixed, however it is computationally expensive. Solution time for the exact calculation of optimal overall cost for the given period assignments also gets computationally inefficient as number of surgeries and/or periods and/or number of scenarios increase. For this reason, we used the most promising sequencing rule and job hedging level from Section 4.2.1 to replace solving problems to optimality for finding sequencing and scheduling decisions.

Initial Population Generation

After population size is determined, that many individuals are generated randomly by assigning a number between 1 and $|T|$ to each surgery for period assignment.

Parent Selection

For selecting parents for crossover, two different methods are tested. First method is choosing parents randomly from the population without considering any property of the parents. In the second method, we assign selection probability to each individual in the population considering their fitness values. The selection probability is calculated as follows:

$$Selection\, probability\; for\; individual\; i = \frac{f_{worst} - f_i}{f_{worst} - f_{best}} \tag{4.2.2}$$

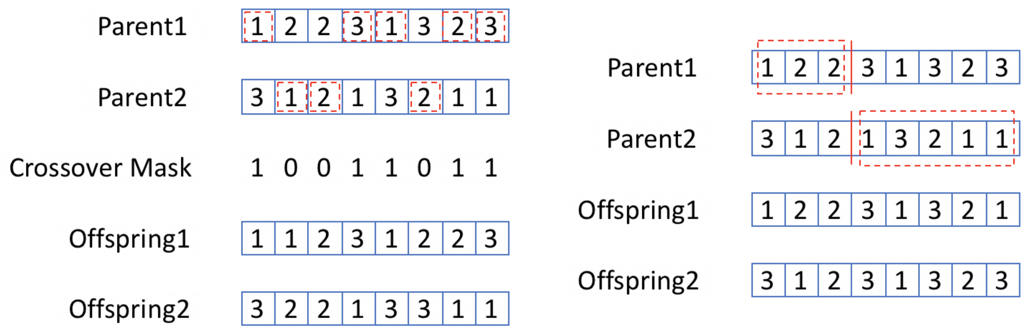
where f_{worst} (f_{best}) is the fitness value of individual with the highest (lowest) cost and f_i is fitness value of i^{th} individual in the population. Then these values are normalized and used as probabilities for selection. By using these probabilities, an empirical distribution function is fitted. Then, two random values between 0 and 1 are generated. The individuals with these cumulative distribution values are selected as parents. Assigning selection probabilities emphasizes the exploitation (intensification) through parent selection which concentrates on better individuals and the offsprings they will produce.

Crossover Operator

In our algorithm, crossover is applied to all pairs of parents selected. Two different crossover operators are tested, (i) Uniform crossover, (ii) 1-point crossover. In uniform crossover, with equal probability, it is decided that which gene will be taken from which parent by using a crossover mask. In 1-point crossover, a cut point is chosen randomly among the genes. Until that point, genes are taken from one parent and rest of the genes are taken from the other parent. Example of both operators are given in Figure 4.6:

Mutation Operator

Mutation operator is important to promote diversity in the population. It helps to explore different parts of the solution space where it is hard to obtain through recombination of parents. We tested two different mutation operators. In the first operator, each gene can be mutated by a predetermined probability p_m. If mutation happens,



(a) Uniform crossover example (b) 1-point crossover example

Figure 4.6: Crossover operators tested in the GA

the day assignment of the surgery is replaced with a randomly chosen period, different than the initial period. In the other operator, with a probability p'_m, the chromosome will undergo mutation, and which gene to be mutated is randomly selected.

Formation of Next Generation

After forming two new offsprings, we have tested three different methods for changing the population and forming the next generation. In the first method, the parent with the worst fitness value is replaced with the best offspring, even though fitness value of offspring is worse. In the second method, an offspring replaces a parent if it has a better fitness value. In the last method, both offsprings are added to the population and worst two individuals among whole population are deleted. When these methods are compared in the sense of exploration and exploitation, first method is focusing on exploration more since worse offsprings can replace better parents. However, the last method emphasizes more on exploitation and faster convergence to a final solution since it always eliminate worst individuals in the population.

Stopping Criteria and Final Solution

When the best fitness value among the population does not improve for a predetermined number of iterations, the algorithm stops. For the individual with the best fitness value, surgery-to-period decisions are fixed in the model and it is solved optimally for determining the final sequences and appointment times.

CHAPTER 5

COMPUTATIONAL EXPERIMENTS

5.1 Results for Exact Solution Methods

In our preliminary experiments, initially, we solved the Extensive Formulation (EF) of APSP by including alternative valid inequalities and symmetry breaking constraints, which are explained in Chapter 3, in order to select the model with best computational performance. After finalizing our model, we solved it by different solution methods explained in Chapter 4 by using data collected from an OPC, presented in [7]. As stated in [7], log-normal distribution fits best for procedure times. For generating surgery durations, *logrnd()* function in MATLAB is used. For the cost parameters (cw, co and cu), values used in [30] are chosen. These values are reported to be estimated by administrators of an OPC. dt is determined by calculating the sum of average surgery durations as it is commonly used in literature (e.g. [29, 30, 32]) and equally distributing this sum to each period by dividing it by $|T|$.

We used the procedure time distributions of surgeries in "Ophthalmology" surgical group. Instances are generated for $|N| = 8$ surgeries considering the surgery mix in the surgical group. The problem setting is given in Table 5.1.

The models are coded in C++ and IBM ILOG CPLEX 12.7.1 is used as the solver. For solving the models, Xcode version 10.3 is used in a computer with 3.5 GHz Intel Core i7 with 16 GB LPDDR3 RAM. As the solution time limit, 7200 CPU seconds is used in preliminary experiments. For the above setting, we solved 10 instances and reported their average results (and worst case results for some measures).

Table 5.1: Preliminary experiment setting

$\vert N\vert$	8
$(\mu_{Type1}, \sigma_{Type1}), \vert N_{Type1}\vert$	$(41.63, 16.43), 6$
$(\mu_{Type2}, \sigma_{Type2}), \vert N_{Type2}\vert$	$(77.66, 44.03), 2$
$\vert T\vert$	3
dt	135
cw	1
cs	0.01
co	33
cu	0

5.1.1 Model Selection

APSP is a NP-hard combinatorial optimization problem and we want to find a stronger model for better computational performance to solve larger instances. With this aim, we derived different lower and upper bounds on allocated surgery times, symmetry breaking constraints for eliminating identical solutions and corresponding valid inequalities.

First of all, we compared the effects of lower and upper bounds on allocated surgery time decision variable, x_{kt}. We used the EF given in Section 3.2 as the model and tested all different combinations of bounds given in Section 3.3.1. The following models are compared in terms of computational performance in the setting given above.

$$x_{kt} \leq \sum_{i \in N^r} MX_{ik} \cdot y_{ikt} \qquad k \in K, t \in T \ (Eq.(3.3.2))$$

$$\sum_{l=1}^{k} x_{lt} \leq \sum_{l=1}^{k} \sum_{i=1}^{N} d_i^{max} \cdot y_{ilt} \qquad k \in K, t \in T \ (Eq.(3.3.1b))$$

$$\sum_{l=1}^{k} \sum_{i=1}^{N} d_i^{max} \cdot y_{ilt} \leq x_{lt} \qquad k \in K, t \in T \ (Eq.(3.3.4))$$

Model A: Base model + Eq. (3.3.2)

Model B: Base model + Eq. (3.3.1b)

Model C: Base model + Eq. (3.3.4)

Model D: Base model + Eq. (3.3.2) + Eq. (3.3.1b)

Model E: Base model + Eq. (3.3.2) + Eq. (3.3.4)

Model F: Base model + Eq. (3.3.1b) + Eq. (3.3.4)

Model G: Base model + Eq. (3.3.2) + Eq. (3.3.1b) + Eq. (3.3.4)

Comparing the results in Table 5.2, we chose Model E, which is adding a lower and upper bound on each x_{kt} variable. Computational times (in CPU seconds) of Model D and E are very close, however, Model E has smaller optimality gaps and it reaches to the optimal solution for more problem instances. Since Constraint (3.3.1b) is not improving our computational performance, we can deduce that using a bound on each variable works better than using a bound on sum of variables in this case.

For the next step, different symmetry breaking constraints from Section 3.3.2 are tested. We conducted experiments with Constraints (3.3.5) - (3.3.7). To analyze the effect of these symmetry breaking constraints on solution performance, following models are solved for different number of scenarios.

$$\sum_{k \in K} k \cdot y_{(N+t)kt} \geq \sum_{k \in K} k \cdot y_{(N+t+1)k(t+1)} \qquad t \in T - \{T-1\} \ (Eq.(3.3.6))$$

$$\sum_{k \in K} \sum_{i \in N^r} y_{ikt} \geq \sum_{k \in K} \sum_{i \in N^r} y_{ik(t+1)} \qquad t \in T - \{T-1\} \ (Eq.(3.3.7))$$

$$\sum_{t=1}^{i} \sum_{k \in K} y_{ikt} = 1 \qquad i = 1, \ldots, T-1 \ (Eq.(3.3.5))$$

Model E1: Model E + Eq. (3.3.6)

Model E2: Model E + Eq. (3.3.7)

Model E3: Model E + Eq. (3.3.6) + Eq. (3.3.7)

Model E4: Model E + Eq. (3.3.5)

In the light of the results in Table 5.3, Model E2 performs best, with being almost 10% better than the next best model. In our problem, eliminating identical solutions by considering the number of surgeries in each period performs better than fixing the period assignment of some surgeries. Even though adding Constraint (3.3.6) and Constraint (3.3.7) separately improves the solution time, including them together in

Table 5.2: Computational performance comparison of combinations of lower and upper bounds on x_{kt}

			|S|=25	|S|=40	|S|=50	|S|=55
Base Model	CPU Time		1212	4272	6872	
	Opt. Gap	Avg.	-	-	5.4%	
		Max.	-	-	10.1%	
	# of Unsolved Instances		0	0	7	
Model A	CPU Time		1394	4224	6375	
	Opt. Gap	Avg.	-	-	4.5%	
		Max.	-	-	8.90%	
	# of Unsolved Instances		0	0	7	
Model B	CPU Time		845	4697	7200	
	Opt. Gap	Avg.	-	-	5.4%	
		Max.	-	-	7.9%	
	# of Unsolved Instances		0	0	10	
Model C	CPU Time		1596	4550	6720	
	Opt. Gap	Avg.	-	0.1%	7.9%	
		Max.	-	1.0%	10.4%	
	# of Unsolved Instances		0	0	10	
Model D	CPU Time		1417	4838	6145	6627
	Opt. Gap	Avg.	-	-	2.2%	8.9%
		Max.	-	-	6.4%	12.6%
	# of Unsolved Instances		0	0	4	9
Model E	CPU Time		976	4409	5148	6687
	Opt. Gap	Avg.	-	-	2.6%	7.7%
		Max.	-	-	7.7%	11.5%
	# of Unsolved Instances		0	0	4	8
Model F	CPU Time		1062	4697	5486	7200
	Opt. Gap	Avg.	-	0.5%	4.0%	9.6%
		Max.	-	4.8%	8.3%	13.6%
	# of Unsolved Instances		0	1	6	10
Model G	CPU Time		1283	5155	5968	
	Opt. Gap	Avg.	-	0.7%	2.9%	
		Max.	-	7.4%	6.9%	
	# of Unsolved Instances		0	1	5	

All models are solved for 10 instances

Table 5.3: Computational performance comparison of symmetry breaking constraints

			|S|=55	|S|=75	|S|=100
Model E	CPU Time		6687		
	Opt. Gap	Avg.	7.7%		
		Max.	11.46%		
	# of Unsolved Instances		8		
Model E1	CPU Time		1512	3587	5738
	Opt. Gap	Avg.	-	-	1.8%
		Max.	-	-	13.0%
	# of Unsolved Instances		0	0	2
Model E2	CPU Time		1550	2798	4072
	Opt. Gap	Avg.	-	-	-
		Max.	-	-	-
	# of Unsolved Instances		0	0	0
Model E3	CPU Time		1994	2628	4423
	Opt. Gap	Avg.	-	-	0.5%
		Max.	-	-	4.8%
	# of Unsolved Instances		0	0	1
Model E4	CPU Time		1589	2939	5556
	Opt. Gap	Avg.	-	-	1.7%
		Max.	-	-	14.0%
	# of Unsolved Instances		0	0	2

All models are solved for 10 instances

the model is not beneficial.

For the final step, we tested some valid inequalities which are derived from the symmetry breaking idea in Constraints (3.3.6) and (3.3.7). To understand which valid inequalities improve our solution times and should be included in our model, following models are solved for different number of scenarios and their results are reported in Table 5.4.

$$\sum_{k \in K_t^d} y_{(N+t)kt} = 1 \qquad t \in T$$
$$\sum_{t \in T} \sum_{k \in K_t^r} y_{ikt} = 1 \qquad i \in N^r \ (Eq.\ (3.3.10))$$

$$\sum_{k \notin K_t^d} y_{(N+t)kt} = 0 \qquad t \in T$$
$$\sum_{t \in T} \sum_{i \in N^r} \sum_{k \notin K_t^r} y_{ikt} = 0 \qquad (Eq.\ (3.3.11))$$

$$\sum_{k \in K_t} \sum_{i \in N^r} y_{ikt} \geq pf_t - 1 \qquad t \in T$$
$$\sum_{k \in K_t} \sum_{i \in N^r} y_{ikt} \leq pl_t - 1 \qquad t \in T \ (Eq.\ (3.3.12))$$

$$\sum_{t \in T} \sum_{k \in K_t} k \cdot y_{(N+t)kt} = N + T \qquad (Eq.\ (3.3.13)$$

Model E2-1: Model E2 + Eq. (3.3.10)
Model E2-2: Model E2 + Eq. (3.3.11)
Model E2-3: Model E2 + Eq. (3.3.12)
Model E2-4: Model E2 + Eq. (3.3.13)
Model E2-5: Model E2 + Eq. (3.3.10) + Eq. (3.3.11) + Eq. (3.3.13)
Model E2-6: Model E2 + Eq. (3.3.10) + Eq. (3.3.11) + Eq. (3.3.13) - Eq. (3.2.3a)

Considering the results in Table 5.4, we chose the final model as Model E2-5. It performs the best among the tested models. Even though adding each valid inequality separately improves the model, Constraints (3.3.10) and (3.3.11) make the major

Table 5.4: Computational performance comparison of valid inequalities derived from symmetry breaking constraints

			|S|=115	|S|=150	|S|=200
Model E2	CPU Time		6074		
	Opt. Gap	Avg.	1.2%		
		Max.	9.4%		
	# of Unsolved Instances		2		
Model E2-1	CPU Time		916		
	Opt. Gap	Avg.	-		
		Max.	-		
	# of Unsolved Instances		0		
Model E2-2	CPU Time		847		
	Opt. Gap	Avg.	-		
		Max.	-		
	# of Unsolved Instances		0		
Model E2-3	CPU Time		5870		
	Opt. Gap	Avg.	5.0%		
		Max.	11.1%		
	# of Unsolved Instances		3		
Model E2-4	CPU Time		1611		
	Opt. Gap	Avg.	-		
		Max.	-		
	# of Unsolved Instances		0		
Model E2-5	CPU Time		574	906	2018
	Opt. Gap	Avg.	-	-	-
		Max.	-	-	-
	# of Unsolved Instances		0	0	0
Model E2-6	CPU Time		559	967	2165
	Opt. Gap	Avg.	-	-	-
		Max.	-	-	-
	# of Unsolved Instances		0	0	0

All models are solved for 10 instances

difference. Including them to the model together also works better. We also tested leaving Constraint (3.2.3a) out of the model since Constraint (3.3.10) and (3.3.11) together can compensate for that constraint, however it makes the computational performance little worse than keeping it.

5.1.2 Solution Method Selection

After deciding on the final model, we solved our problem by different solution methods. All methods and their abbreviations used in this section are given in Table 5.5.

Initially, we compared the performances of solving EF, using L-shaped Method (LM), and L-shaped method with mean value cuts (LM-MVC). The solution times are given in Table 5.6. For the instances, for which the considered method cannot reach to the optimal solution within the time limit, average and maximum optimality gap values across 10 instances are reported. We reported two optimality gaps. In the L-shaped algorithm, in every iteration, RMP calculates a lower bound (LB) for the problem since it is a minimization problem. Then, by solving the SPs with given first-stage solutions, an upper bound (UB) is calculated. The algorithm converges to the optimal solution while LB and UB converges to each other. In % Opt. Gap, we used LB in last iteration (highest) and lowest UB among all iterations while in % Opt. Gap*, we used the optimal solution for the problem as UB since we have this value from the extensive model.

$$Opt.Gap* = \frac{Opt.\ Soln. - LB}{Opt.\ Soln.}$$
$$Opt.Gap = \frac{Best\, UB - LB}{Best\, UB}$$

Solving the extensive formulation works well and finds the optimal solution in the given time limit for this setting up to $|S| = 200$ scenarios. It performs best among three solution methods. However, as number of scenarios increase, solution time increases exponentially. Therefore, solving the extensive formulation may not be practical for larger instances with higher number of scenarios. Both LM and LM-MVC failed to solve any of the instances in the given time limit. On the average, LM has proceeded 150 more iterations in the same amount of time then LM-MVC. Even though LM-MVC has to solve larger RMPs, MVC can accelerate the convergence. It

Table 5.5: Summary table for the solution methods tested

Method	**Abbreviation**	**RMP**	**SP**	**SP decomp. based on**	**Embedded CPLEX Benders' Decomp. Alg.**
Extensive Formulation	**EF**	-	-	-	✗
L-shaped Method	**LM**	y_{ikt}, x_{kt}	$P_{ikts}, S_{ikts}, o_{ts}, u_{ts}$	S	✗
L-shaped Method - SP decomposition considering only T	**LM-T**	y_{ikt}, x_{kt}	$P_{ikts}, S_{ikts}, o_{ts}, u_{ts}$	T	✗
Multi-Cut L-shaped Method	**MLM**	y_{ikt}, x_{kt}	$P_{ikts}, S_{ikts}, o_{ts}, u_{ts}$	S	✗
Multi-Cut L-shaped Method - SP decomposition considering both T and S	**MLM-TS**	y_{ikt}, x_{kt}	$P_{ikts}, S_{ikts}, o_{ts}, u_{ts}$	T & S	✗
Multi-Cut L-shaped Method - SP decomposition considering only T	**MLM-T**	y_{ikt}, x_{kt}	$P_{ikts}, S_{ikts}, o_{ts}, u_{ts}$	T	✗
B&C Benders' Decomposition Algorithm	**B&C BDA**	y_{ikt}	$x_{kt}, P_{ikts}, S_{ikts}, o_{ts}, u_{ts}$	T	✗
CPLEX Automatic Benders' Decomposition Algorithm - Full	**A-BDA-Full**	y_{ikt}	$x_{kt}, P_{ikts}, S_{ikts}, o_{ts}, u_{ts}$	T	✓
CPLEX Automatic Benders' Decomposition Algorithm - L-shaped Method	**A-BDA-LM**	y_{ikt}, x_{kt}	$P_{ikts}, S_{ikts}, o_{ts}, u_{ts}$	S	✓
CPLEX Automatic Benders' Decomposition Algorithm - L-shaped Method - SP decompostion considering both T and S	**A-BDA-LM-TS**	y_{ikt}, x_{kt}	$P_{ikts}, S_{ikts}, o_{ts}, u_{ts}$	T & S	✓
CPLEX Automatic Benders' Decomposition Algorithm - L-shaped Method - SP decompostion considering only T	**A-BDA-LM-T**	y_{ikt}, x_{kt}	$P_{ikts}, S_{ikts}, o_{ts}, u_{ts}$	T	✓

also provides a better initial solution for θ, approximate value of the expected cost of second-stage problem. When optimality gaps are compared, there is not a significant difference between both methods to decide on a superior one. This means, through the iterations, they reach to very close solutions (UBs).

Table 5.6: Solution time comparison between solving EM, using LM and LM-MVC for three different values of $|S|$

	EF	LM					LM-MVC				
	Soln. Time (CPU sec.)	% Opt. Gap* Avg.	% Opt. Gap* Max.	% Opt. Gap Avg.	% Opt. Gap Max.	# of iter.	% Opt. Gap* Avg.	% Opt. Gap* Max.	% Opt. Gap Avg.	% Opt. Gap Max.	# of iter.
|S|=50	84.1	8.30%	11.35%	10.81%	14.65%	863	8.81%	11.98%	11.65%	17.12%	698
|S|=100	351.9	9.53%	11.95%	12.08%	15.90%	835	9.95%	12.87%	12.54%	16.36%	705
|S|=200	2017.7	12.69%	15.61%	15.87%	19.01%	798	12.71%	14.84%	15.93%	19.37%	649

EF managed to solve all five instances while LM and LM-MVC failed to solve any of the instances

In LM, only a single optimality cut is added to RMP in each iteration. When compared with the multi-cut Lshaped Method (MLM), less information could be transferred to RMP during iterations. There is a trade-off between solving larger problems for better solutions in each iteration and solving smaller problems to process more iterations. We compared single-cut and multi-cut L-shaped algorithms in terms of their computational performances. Results are given in the "Iterative L-shaped" column of Table 5.7. Even though effect of MVC is not clear in the iterative L-shaped method, in the multi-cut version, they improve computational performance of the method clearly. Its effect also increases with higher number of scenarios.

For our problem, adding more cuts and carrying more information to RMP in each iteration brings a computational advantage. Solving larger RMPs for fewer iterations outperforms solving smaller RMPs and making more iterations. While LM-MVC cannot find the optimal solution in the time limits, MLM-MVC and MLM with subproblems which are decomposed based on periods and scenarios with MVC (MLM-TS-MVC) finds optimal solutions in a reasonable amount of time (Detailed explanations of these methods are given in Chapter 4). Also, MLM-TS-MVC outperforms MLM-MVC in all three $|S|$ values. Even though MLM-TS-MVC may generate $|T|$ times more optimality cuts per iteration, when the total number of cuts added to RMP is compared, MLM-MVC required more optimality cuts. MLM-MVC needs to make

Table 5.7: Computational results of multi-cut L-shaped methods

		Iterative L-shaped				B&C based L-shaped		
		LM-MVC	MLM	MLM-MVC	MLM-TS-MVC	LM-MVC	MLM-MVC	MLM-TS-MVC
$\lvert S\rvert$=50	Soln. Time	7200 (11.65%)	411.9	385.1	104.4	872.8	16.6	7.9
	# of Iteration	698	27	23	12	9481	50	31
	# of Cuts	698	989	753	669	9481	988	730
$\lvert S\rvert$=100	Soln. Time	7200 (12.54%)	1358.6	994.7	290.4	2249	43.2	18.8
	# of Iteration	705	28	23	13	12855	53	32
	# of Cuts	705	1957	1460	1339	12855	1849	1424
$\lvert S\rvert$=200	Soln. Time	7200 (15.93%)	6449.8	5796.1	1422.7	6845.3	174.1	65.6
	# of Iteration	649	32	27	16	10451	61	53
	# of Cuts	649	4076	3201	2771	10451	3883	2834

Solution times are reported in CPU seconds.

Iterative LM-MVC failed to solve any of the instances in the time limit, optimality gaps are stated in parenthesis.

For 200 scenarios, B&C LM-MVC failed to solve 3 instances out of 10 in the time limit with optimality gap value less than 10%.

1.5 to 2 times iterations of MLM-TS-MVC to reach optimality.

In the branch-and-cut framework (B&C), different than the iterative LM, only a single branch-and-bound tree is generated, and optimality cuts are added at each incumbent solution as lazy constraints. Results of using variants of the L-shaped algorithm in B&C framework can be found in Table 5.7. Solving the problem in B&C framework accelerated all the methods drastically.

For implementing BDA, we used the pre-defined Automatic Benders' decomposition algorithm (A-BDA) available in CPLEX. This feature is introduced in CPLEX 12.7. In this method, with default parameter setting (A-BDA-Full), RMP includes period assignment/sequencing decision variables only (leaving time allocation decision to SP). As a result, RMP becomes easier to solve whereas SPs got much larger with considering all scenario based variables at once and additional x_{kt} variables. However, SPs are still LP and easy to solve. Detailed explanation of BDA is given in 4.1.2.4. BDA outperforms all solution methods in all instances as it is given in Table 5.8. In small number of scenarios ($|S| < 200$), B&C based MLM-TS-MVC performs similar to A-BDA-Full, however, in $|S| = 200$, the computational time difference is clear.

CPLEX is a commercial solver and they do not reveal the details of their algorithm. Results of solution methods using A-BDA are given in Table 5.8. A-BDA uses not only standard Benders' cut but also other cuts (e.g. flow cuts, mixed integer rounding cuts) to accelerate the algorithm as it is reported in the log files of the solutions. Also, we believe they are implementing the algorithm in a B&C framework since iterative methods are dominated by B&C used methods. To better understand the dynamics of the automatic algorithm, we mimicked the decomposition of A-BDA-Full by B&C framework, through lazy constraint callback structure in CPLEX (which we have used for prior B&C based L-shaped methods) and moved x_{kt} decision variables to second-stage problem (B&C BDA). We also tested the effect of MVC in this algorithm, even though MVC are generated considering the mean value scenario and here the decomposition is based on periods. Using B&C BDA with MVC also solved instances in shorter time than without MVC. A-BDA-Full outperforms B&C BDA and this can be due to additional cuts added in A-BDA and/or due to some unknown pre-processing of the problem. Then, we compare A-BDA-Full with MLM-T-MVC, to see how having surgery duration allocation variable in first-stage problem will effect. A-BDA-Full performs a lot better than MLM-T-MVC and they are incomparable. MLM-T-MVC also performs the worst among methods using multi-cut L-shaped methods as it can be seen in "B&C based L-shaped" columns of Table 5.8.

MVC is designed for scenario based decomposition; however, A-BDA-Full has a different decomposition structure. In A-BDA, MVC leads to larger and more complex RMP. It has negative effect on the computational performance of the algorithm and sometimes makes CPLEX give error of "unable to solve MIP" during the algorithm. Due to these reasons, we excluded MVC from automatic BDA experiments. The automatic BDA performs better when the RMP is pure IP.

As the second part of our comparison, we mimicked the decomposition of L-shaped method and its variants in the A-BDA. As defined in Chapter 4, L-shaped method is a special case of Benders' algorithm. When results in "CPLEX Automatic Benders' Decomposition Algorithm" and "B&C based L-shaped" columns in Table 5.8 are compared, the resulting solution times are closer to B&C framework. However, except MLM-T-TS, B&C MLM-MVC and MLM-TS-MVC are performing better in all instances. These results support our opinion about using A-BDA with a pure IP

RMP works better.

After a detailed computational performance of the solution methods, we decided to continue with A-BDA-Full for our main experiments since it has the best computational performance among all other methods.

5.2 Results for Heuristic Solution Methods

5.2.1 Insights from Simple Sequencing Rules

APSP is a complex problem which includes three different decisions: period assignment, sequencing and scheduling. It is important to understand the impact of each of these decisions in the constructed schedules for gaining better insight about the problem. For this purpose, we used simple sequencing heuristics explained in Section 4.2 for finding out well performing structures. We used the instances in preliminary experiments, having the setting given in Table 5.1 with $|S| = 200$.

Initially, period assignment (PA) decision is considered. By all possible PA decisions, using simple sequencing rules, surgery-to-period assignments are made. Then, for each period, surgeries are sequenced and surgery durations are allocated optimally. This setting is represented by "PA/*/*" and optimal solution is represented by "*/*/*". In Table 5.9, results are given. Results stated in the table are average and maximum values of 10 instances.

LPT-p50/*/* performs best and CoefA-μ/*/* and VarD-p50/*/* have close performance to that. Also their solution times are under 1.5 seconds. The best solution we could achieve by these rules is 7.54% on the average. In LPT, surgery time is tried to be distributed evenly to the periods and in VarD and CoefD, uncertainty and variance of surgery durations are distributed evenly to the periods. Since overtime cost dominates the waiting time cost in our considered setting, it is more important to complete surgeries in available regular time. For the next step, we used same six sequencing rules for sequencing surgeries in each period.

For period assignment, we used promising LPT-p50, CoefD-μ and VarD-p25 for pe-

Table 5.8: Computational results of solution methods using branch and bound framework and embedded Benders' algorithm in CPLEX

		B&C based L-shaped				B&C based Benders' Algorithm		CPLEX Automatic Benders' Decomposition Algorithm			
		LM-MVC	MLM-T-MVC	MLM-MVC	MLM-TS-MVC	w/o MVC	w/ MVC	A-BDA-Full	A-BDA-LM-S	A-BDA-LM-TS	A-BDA-LM-T
|S|=50	Soln. Time	872.8	596.4	16.6	7.9	97.9	96.3	6.4	26.1	13.9	16.0
	# of Cuts	9481	967	50	31	508	472	314	617	330	769
|S|=100	Soln. Time	2249	2597	43.2	18.8	285.4	266.0	13.7	90.7	41.7	26.7
	# of Cuts	12855	1126	53	32	542	507	368	1103	668	957
|S|=200	Soln. Time	6845.3	7200 (25.65%)	174.1	65.6	795.7	742.8	26.3	364.5	179.5	51.9
	# of Cuts	10451	945	61	53	894	840	461	2520	1359	1361

Solution times are reported in CPU seconds.

For 200 scenarios, B&C LM-MVC failed to solve 3 instances out of 10 in the time limit with optimality gap value less than 10%.

For A-BDA, number of Benders' Cuts are reported.

Table 5.9: Results of using sequencing rules for PA

	Obj. Func. Val	Soln. Time (CPU sec.)	% Opt. Gap Average	% Opt. Gap Max
//*	1644.90	26.34	-	-
$SPT-\mu/*/*$	2233.29	1.30	35.92%	42.18%
$SPT-p25/*/*$	2232.96	1.26	35.87%	41.16%
$SPT-p50/*/*$	2237.67	1.29	36.17%	42.18%
$SPT-p75/*/*$	2243.94	1.33	36.55%	43.16%
$LPT-\mu/*/*$	1772.53	1.30	7.87%	11.75%
$LPT-p25/*/*$	1784.66	1.27	8.55%	11.16%
$LPT-p50/*/*$	**1767.73**	**1.30**	**7.54%**	**11.64%**
$LPT-p75/*/*$	1786.88	1.29	8.69%	12.74%
$VarA-\mu/*/*$	2234.75	1.28	36.03%	43.06%
$VarA-p25/*/*$	2233.41	1.24	35.95%	42.19%
$VarA-p50/*/*$	2241.54	1.23	36.41%	43.06%
$VarA-p75/*/*$	2247.38	1.18	36.74%	42.31%
$VarD-\mu/*/*$	1757.32	1.31	6.96%	11.37%
$VarD-p25/*/*$	1778.83	1.29	8.24%	12.26%
$VarD-p50/*/*$	**1768.17**	**1.29**	**7.62%**	**11.37%**
$VarD-p75/*/*$	1774.58	1.30	7.94%	14.27%
$CoefA-\mu/*/*$	2231.51	1.23	35.83%	43.38%
$CoefA-p25/*/*$	2240.25	1.21	36.35%	43.38%
$CoefA-p50/*/*$	2238.73	1.22	36.25%	43.38%
$CoefA-p75/*/*$	2253.51	1.20	37.11%	42.73%
$CoefD-\mu/*/*$	**1770.12**	**1.22**	**7.72%**	**13.13%**
$CoefD-p25/*/*$	1788.32	1.33	8.79%	15.16%
$CoefD-p50/*/*$	1776.43	1.25	8.10%	13.13%
$CoefD-p75/*/*$	1777.39	1.21	8.10%	13.13%

Table 5.10: Results of using sequencing rules for surgery sequencing while PA decisions are fixed

	Obj. Func. Val	Soln. Time (CPU sec.)	% Opt. Gap	
			Average	Max
//*	1644.9	26.34	-	-
$LPT-p50/*/*$	1767.73	1.30	7.54%	11.64%
$\boldsymbol{LPT-p50/SPT/*}$	**1770.28**	**0.17**	**7.69%**	**11.99%**
$LPT-p50/LPT/*$	1815.51	0.16	10.45%	14.48%
$\boldsymbol{LPT-p50/VarA/*}$	**1770.89**	**0.16**	**7.73%**	**11.93%**
$LPT-p50/VarD/*$	1815.95	0.16	10.48%	14.59%
$\boldsymbol{LPT-p50/CoefA/*}$	**1770.31**	**0.17**	**7.69%**	**11.93%**
$LPT-p50/CoefD/*$	1815.07	0.16	10.42%	14.59%
$VarD-p50/*/*$	1768.17	1.29	7.62%	11.37%
$\boldsymbol{VarD-p50/SPT/*}$	**1770.75**	**0.16**	**7.78%**	**11.72%**
$VarD-p50/LPT/*$	1814.52	0.16	10.45%	15.24%
$\boldsymbol{VarD-p50/VarA/*}$	**1771.42**	**0.17**	**7.82%**	**11.80%**
$VarD-p50/VarD/*$	1814.12	0.17	10.42%	15.02%
$\boldsymbol{VarD-p50/CoefA/*}$	**1770.81**	**0.16**	**7.78%**	**11.80%**
$VarD-p50/CoefD/*$	1814.17	0.16	10.43%	15.02%
$CoefD-\mu/*/*$	1770.12	1.22	7.72%	13.13%
$\boldsymbol{CoefD-\mu/SPT/*}$	**1773.63**	**0.16**	**7.93%**	**13.36%**
$CoefD-\mu/LPT/*$	1818.39	0.15	10.64%	15.76%
$\boldsymbol{CoefD-\mu/VarA/*}$	**1774.23**	**0.16**	**7.96%**	**13.28%**
$CoefD-\mu/VarD/*$	1818.80	0.16	10.67%	16.21%
$\boldsymbol{CoefD-\mu/CoefA/*}$	**1773.50**	**0.15**	**7.92%**	**13.28%**
$CoefD-\mu/CoefD/*$	1818.21	0.16	10.63%	16.21%

riod assignment and SPT, VarA and CoefA for sequencing. Then the problem of surgery time allocation is solved optimally. The results are given in Table 5.10. For all of the sequencing rules, totally different than PA decision, SPT, VarA and CoefA observed to be working well. The rules which place surgeries with lower variance or shorter surgery times are performing better. For example VarA is a commonly used rule for sequencing in the literature. The main idea here is to prevent accumulation of delays for surgeries in later part of the period.

For the last decision, surgery time assignment, we used four different job hedging levels. Results of these schedules are given in Table 5.11.

From Table 5.11, we see that 25^{th} percentile works best as the job hedging level. The surgery times are generated from a log-normal distribution and log-normal distribution is a right-skewed distribution. For this reason, for better estimating the job hedging level, $25^{th}\%$ percent seems reasonable. However, the effect of cost parameters is also important. Since $\frac{co}{cw} = 33$, overtime is more important than waiting time. This will lead allocation of shorter surgeries times, ignoring some of the variation in the surgery durations. This can be another reason for 25^{th} percentile working well. Among all decisions, the biggest source of the optimality gap for $\frac{co}{cw} = 33$ case is the period assignment decision. Better rules for PA decision can bring more benefit compared to S and DA.

5.2.2 Parameter Selection for Genetic Algorithm

We developed a Genetic Algorithm for the APSP, which is defined in Section 4.2.3. To determine the algorithm parameters, we have tested different population sizes, crossover and mutation operators, parent selection rules and ways of generating next generation. Table of tested experiment settings are given in Table 5.12.

We tested our GA on the preliminary problem setting given in Table 5.1 with $|S| = 200$. We tested our algorithm on 10 instances and run our algorithm for 10 times for each instance. Detailed computational results are given in Appendix C. Average of 10 replications for 10 instances are reported. As a stopping criteria, 2000 iterations are selected to not miss any improvement. As it can be seen from the table, the main part

Table 5.11: Results of four different job hedging levels for surgery time allocation when planning decisions are fixed

	Obj. Func. Val	Soln. Time (CPU sec.)	Opt. Gap Average	Opt. Gap Max
//*	1644.9	26.34	-	-
$LPT - p50/SPT/*$	1770.28	0.17	7.69%	11.99%
$LPT - p50/SPT/\mu$	2024.58	0.10	23.21%	27.68%
$LPT - p50/SPT/p25$	1790.95	0.10	8.95%	12.72%
$LPT - p50/SPT/p50$	1918.83	0.10	16.73%	19.59%
$LPT - p50/SPT/p75$	2575.19	0.11	61.09%	69.92%
$LPT - p50/VarA/*$	1770.89	0.16	7.73%	11.93%
$LPT - p50/VarA/\mu$	2023.36	0.11	23.14%	26.11%
$\boldsymbol{LPT - p50/VarA/p25}$	**1790.21**	**0.10**	**8.91%**	**12.86%**
$LPT - p50/VarA/p50$	1923.06	0.10	17.00%	20.40%
$LPT - p50/VarA/p75$	2571.43	0.10	64.03%	73.84%
$LPT - p50/CoefA/*$	1770.31	0.17	7.69%	11.93%
$LPT - p50/CoefA/\mu$	2023.99	0.10	31.70%	38.73%
$LPT - p50/CoefA/p25$	1790.31	0.11	18.89%	27.58%
$LPT - p50/CoefA/p50$	1921.64	0.13	30.29%	38.81%
$LPT - p50/CoefA/p75$	2572.93	0.11	78.19%	89.13%
$VarD - p50/SPT/*$	1770.75	0.16	7.78%	11.72%
$VarD - p50/CoefA/\mu$	2017.77	0.10	31.30%	40.93%
$VarD - p50/CoefA/p25$	1789.27	0.10	18.82%	28.59%
$VarD - p50/CoefA/p50$	1929.02	0.11	30.79%	40.49%
$VarD - p50/CoefA/p75$	2563.26	0.11	77.51%	88.81%
$VarD - p50/VarA/*$	1771.42	0.17	7.82%	11.80%
$VarD - p50/SPT/\mu$	2021.59	0.10	23.03%	26.73%
$VarD - p50/SPT/p25$	1789.97	0.10	8.94%	12.61%
$VarD - p50/SPT/p50$	1925.88	0.10	17.18%	20.71%
$VarD - p50/SPT/p75$	2572.73	0.11	60.93%	70.42%
$VarD - p50/CoefA/*$	1770.81	0.16	7.78%	11.80%
$VarD - p50/VarA/\mu$	2017.03	0.10	22.76%	27.07%
$\boldsymbol{VarD - p50/VarA/p25}$	**1789.38**	**0.10**	**8.91%**	**12.68%**
$VarD - p50/VarA/p50$	1930.07	0.10	17.45%	21.31%
$VarD - p50/VarA/p75$	2559.81	0.11	63.28%	75.05%
$CoefD - \mu/SPT/*$	1773.63	0.16	7.93%	13.36%
$CoefD - \mu/SPT/\mu$	2017.75	0.11	22.81%	30.61%
$CoefD - \mu/SPT/p25$	1791.37	0.12	9.01%	14.28%
$CoefD - \mu/SPT/p50$	1928.56	0.10	20.64%	28.81%
$CoefD - \mu/SPT/p75$	2557.45	0.11	69.84%	80.72%
$CoefD - \mu/VarA/*$	1774.23	0.16	7.96%	13.28%
$CoefD - \mu/VarA/\mu$	2015.38	0.12	22.66%	29.82%
$CoefD - \mu/VarA/p25$	1794.10	0.11	9.18%	14.64%
$CoefD - \mu/VarA/p50$	1929.09	0.11	23.05%	31.57%
$CoefD - \mu/VarA/p75$	2556.04	0.10	73.30%	85.43%
$CoefD - \mu/CoefA/*$	1773.50	0.15	7.92%	13.28%
$CoefD - \mu/CoefA/\mu$	2014.78	0.11	22.63%	29.82%
$CoefD - \mu/CoefA/p25$	1793.77	0.10	9.16%	14.64%
$CoefD - \mu/CoefA/p50$	1927.41	0.11	25.42%	34.21%
$CoefD - \mu/CoefA/p75$	2554.47	0.10	76.91%	88.82%

Table 5.12: Parameters for Genetic Algorithm to be tested

Algorithm Parameters	Alternatives to be tested
Popsize	30
	100
Crossover	***Uniform Crossover***
	1-point Crossover
Mutation	***M1: Each gene can be mutated w/ prob. 0.001***
	M2: Random gene from a chromosome is mutated w/ prob 0.005
Parent Selection	***Parents are randomly chosen w/ equal prob.***
	Selection prob. Proportional to fitness values
Forming Next Gen.	wpbo: Worst Parent is exchanged with best offspring
	2of4: Parents are deleted from population and best 2 out of 4 (2 parents 2 offsprings) are added
	worst2: Both offsprings added to population and 2 worst individuals are deleted from population

of the solution time is finding the final optimal solution for the best period assignment. Iterations take less than half second. For the solution of the final problem, we used A-BDA-Full for improving our computational performance. For the fitness function, we used the best performing sequencing and scheduling rules which are VarA for sequencing surgeries and $p25$ for the job hedging.

To choose the best algorithm factors, we conducted and analyzed the full factorial design of experiment of the parameters. In the analysis average 10 replication for each instance for each parameter setting is used. We have $2 \times 2 \times 2 \times 2 \times 3 = 48$ parameter settings to test. In Minitab 17, we created Main Effect and Interaction plots of each factor for the average optimality gap. From the plot in Figure 5.1., we see that using a bigger population with uniform crossover ends up in smaller average gap values. For forming the next generation, first deleting the chosen parents from the population and adding best two individuals after crossover works best. When compared with the other ways of forming the next generation, its position is in between exploration and exploitation. It does not allow population to have worse individuals however still there can be worse individuals than the offsprings or parents chosen which are not replaced inthe population. In the main effects plot, mutation operator and parent selection rule seems indifferent, so we also checked main effects plot of number of time optimal solution is reached. Figure 5.2 is the main effect plot for the number of times optimal solution is found and higher values are better this time. It shows parallel behavior in terms on effects of factors. Still effect of mutation and parent selection is not critical, we select M1 for the mutation operator and random parent selection rule.

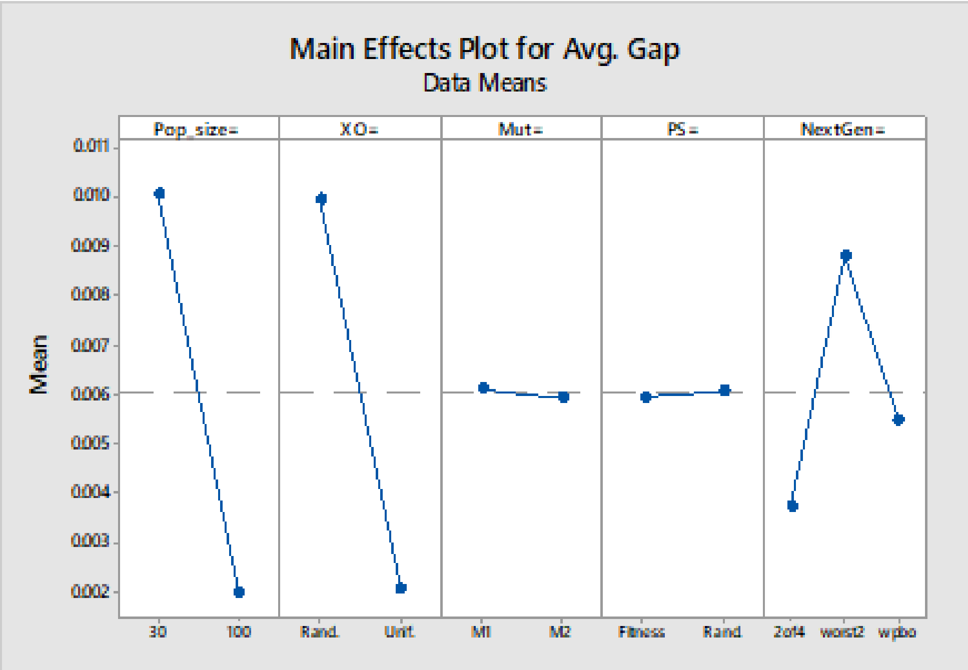


Figure 5.1: Main effect plot for average gap for all factors

Then we checked the interaction plot for average gap which is given in Figures 5.3. There is only interaction seen between forming next generation and parent selection and next generation and mutation. When interaction of these factors are checked by 2-way ANOVA, we see that it is not important.

For the final parameter setting, we chose population size of 100, uniform crossover operator, mutation operator M1, random parent selection for crossover and 2of4. For further experiments, we use this setting for GA.

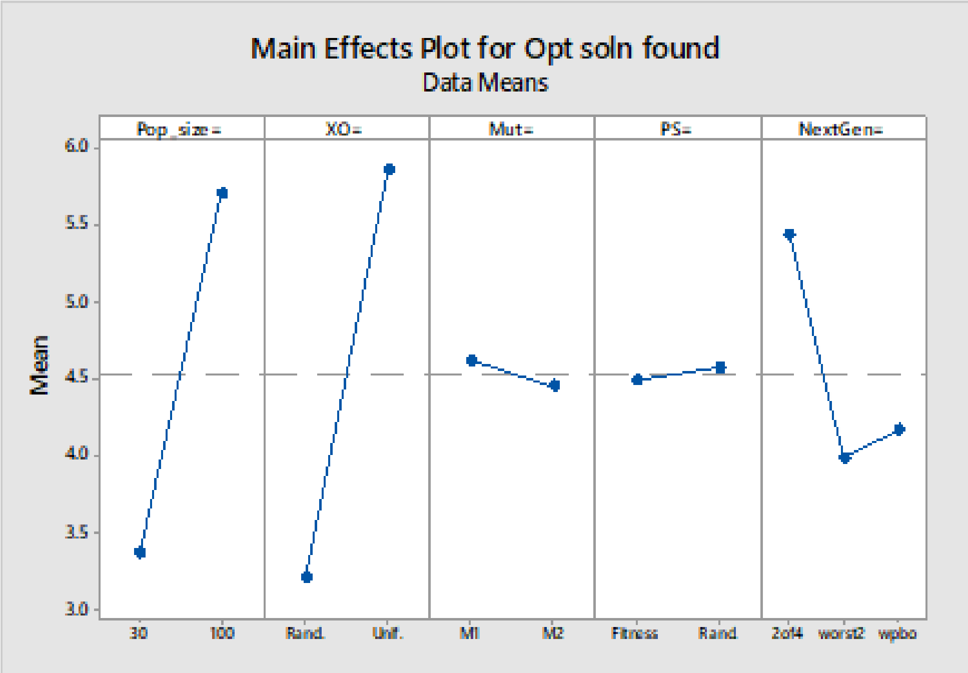


Figure 5.2: Main effect plot for number of time the optimal solution is found for all factors

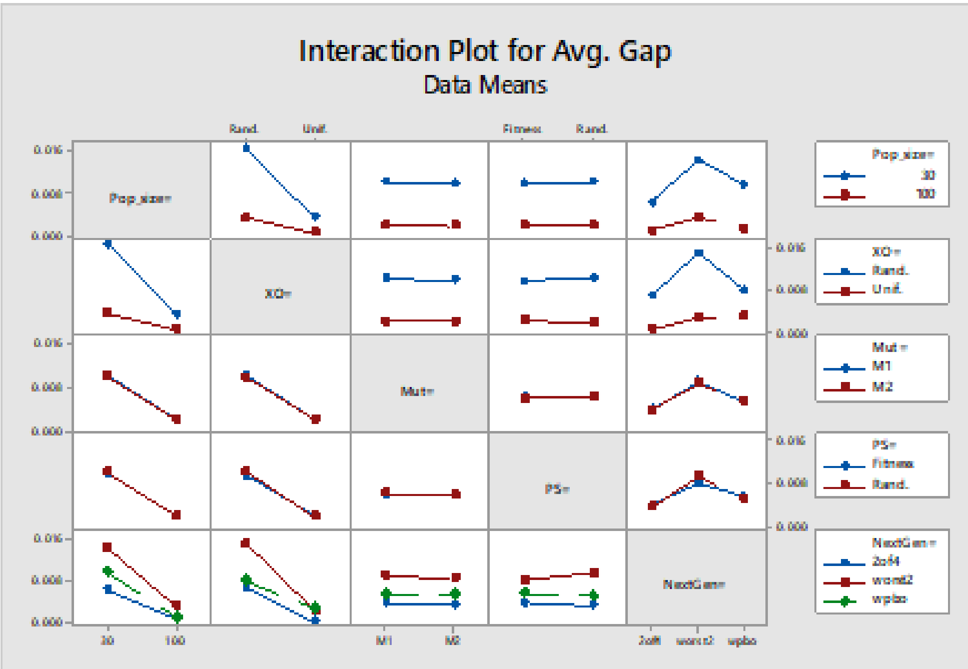


Figure 5.3: Interaction plot for average gap for all factors

Table 5.13: New experiment setting

$\lvert N \rvert$	10
$(\mu_{Type1}, \sigma_{Type1})$, $\lvert N_{Type1} \rvert$	$(20.49, 10.86), 2$
$(\mu_{Type2}, \sigma_{Type2})$, $\lvert N_{Type2} \rvert$	$(20.93, 15.08), 5$
$(\mu_{Type3}, \sigma_{Type3})$, $\lvert N_{Type3} \rvert$	$(34.01, 17.42), 3$
$\lvert T \rvert$	3
dt	82.55
cw	1
cs	0.01
cu	0

5.3 Experiments for Managerial Insights

For the further experiments, we chose our surgery mix from "Pain Medicine" surgical group of OPC given in [7]. Procedure groups 2, 3 and 5 are used since they include 97% of the total operations for that group. Total number of surgeries is increased to $N = 10$. Problem setting is given in Table 5.13.

In our model, surgery durations are random variables with known distributions. We represent these distributions through scenarios, realizations of these random variables, to consider uncertainties in our model. To reach the "true optimum" for the "original problem", it is significant to choose correct number of scenarios. Using higher number of scenarios represents distributions better, however, problem should stay tractable to reach the optimal solution. To determine the minimum number of scenarios for our problem, we solved 10 instances from the setting given in Table 5.13 up to 1500 scenarios. We reached 1500 scenario case by adding 100 new scenarios every time we increase the number of scenarios. For solving the models, we used the best performing method, A-BDA-Full, which is determined in the previous section. The objective function values for 10 instances up to 1500 scenarios is given in Figure 5.4.

We observe that in the small number of scenarios, the variation in the objective function value is high, however, it decreases as $\lvert S \rvert$ increases. In Table 5.14, average of

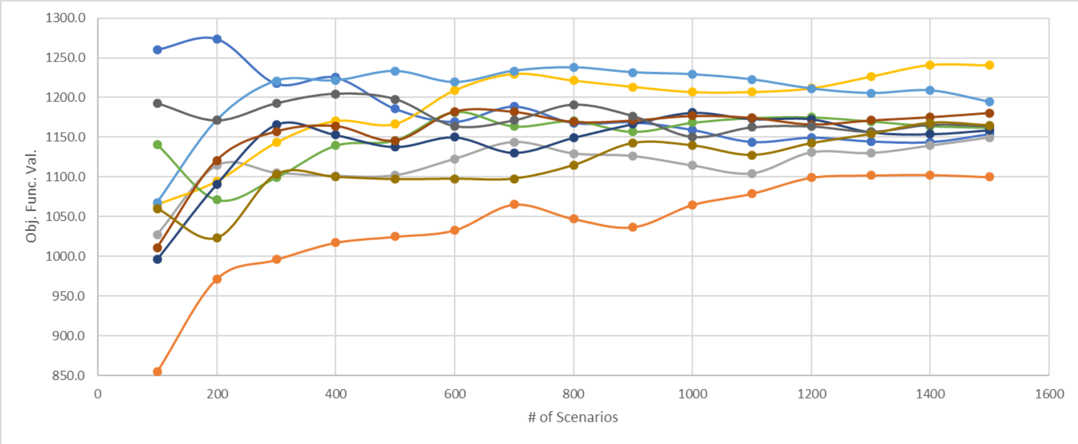


Figure 5.4: Objective function value of 10 instances up to $|S| = 1500$

percentage difference between objective function values of two consecutive cases ($|S|$ and $|S| + 100$) are given.

When Figure 5.4 and Table 5.14 are considered, we decided to use $|S| = 800$ for our main experiments. The change between consecutive scenario cases after $|S| = 800$ are less than %1 for almost all values. Also in the graph we see more stabilized lines after 800 scenarios which implies that it starts to converge to "optimal solution" for the "original problem".

5.3.1 Analysis of the Problem Setting

We solved our problem with three different cost parameter settings, $\frac{co}{cw} = 33$, $\frac{co}{cw} = 10$ and $\frac{co}{cw} = 1$. $\frac{co}{cw} = 33$ is the case used in [30], which is closest to the real practice in the OPC they studied. In this case, OR overtime is significantly more important than the patient waiting time. In $\frac{co}{cw} = 10$ case, OR overtime is still more expensive but patient waiting times are more important compared to the previous setting. In our last case, we consider $\frac{co}{cw} = 1$, where patient waiting time and OR overtime are equally important. Case $\frac{co}{cw} \leq 1$, where patient waiting time is significantly more important, is not considered since it is not used in practice. Results of each case for ten different instances generated from same setting are given in Table 5.15. Solution time of each scenario is reported with average expected waiting time ($E[W]$), idle time ($E[I]$) and overtime ($E[O]$) per period.

Table 5.14: Average percentage difference between two cases with 100 scenario difference

# of Scenarios	Avg. % Difference
100	6.74%
200	3.88%
300	1.22%
400	0.96%
500	1.91%
600	1.34%
700	1.22%
800	0.90%
900	1.05%
1000	0.76%
1100	0.83%
1200	0.63%
1300	0.55%
1400	0.48%
1500	-

$$E[W] = \frac{1}{|S| \times |T|} \cdot \sum_{s \in S} \sum_{t \in T} \sum_{k \in K_t} \sum_{i \in N^r} p^s_{ikt}$$
$$E[I] = \frac{1}{|S| \times |T|} \cdot \sum_{s \in S} \sum_{t \in T} \sum_{k \in K_t} \sum_{i \in N^r} s^s_{ikt}$$
$$E[O] = \frac{1}{|S| \times |T|} \cdot \sum_{s \in S} \sum_{t \in T} o^s_t$$

For all instances, as $\frac{co}{cw}$ decreases, $E[W]$ per period decreases and $E[I]$ per period and $E[O]$ per period increases since importance of patient waiting increases as it is given in Table 5.15. However, the main increase is in idle time, even though we do not change cs. The main reason is, for decreasing the patient waiting times, longer durations are allocated to surgeries to avoid patient waiting and this causes idle time. When $\frac{co}{cw} = 33$, OR utilization is the highest and the idle time is 1.9% of the session length. $E[W]$ per period is almost 3.5 times of $E[O]$ per period in this case. We can see for the highest overtime cost setting that, on the average, there is extra 12.9% overtime for each period. When $\frac{co}{cw} = 10$, in the case of patient waiting and OR

Table 5.15: Results of each instance for three different values of overtime cost over waiting time cost parameters

		Avg	Ins 1	Ins 2	Ins 3	Ins 4	Ins 5	Ins 6	Ins 7	Ins 8	Ins 9	Ins 10
	Soln. Time (CPU sec.)	**4123.97**	4229.79	4328.55	3654.75	4253.67	4112.03	3770.25	4173.47	4508.32	4141.48	4067.39
	E[W] per period (min)	**35.85**	35.78	35.76	35.79	34.91	35.55	37.55	36.49	35.40	33.98	37.34
$\frac{co}{cw} = 33$	Avg. Patient Waiting (min)	**10.76**	10.73	10.73	10.74	10.47	10.67	11.27	10.95	10.62	10.19	11.20
	E[I] per period (min)	**1.59**	1.48	1.48	1.63	1.61	1.53	1.79	1.40	1.87	1.59	1.52
	E[O] per period (min)	**10.64**	10.27	10.93	10.99	9.98	10.75	11.33	11.08	10.42	9.70	10.98
	Soln. Time (CPU sec.)	**11088.31**	11645.00	9934.74	9304.76	12112.10	8978.17	9654.63	11205.10	12320.80	12468.60	13259.20
	E[W] per period (min)	**27.01**	26.94	27.19	26.29	26.71	27.14	27.93	27.29	27.02	25.54	28.02
$\frac{co}{cw} = 10$	Avg. Patient Waiting (min)	**8.10**	8.08	8.16	7.89	8.01	8.14	8.38	8.19	8.11	7.66	8.40
	E[I] per period (min)	**3.77**	3.76	3.61	3.85	3.82	3.61	3.89	3.63	3.81	3.93	3.81
	E[O] per period (min)	**11.16**	10.78	11.44	11.56	10.49	11.22	11.88	11.61	10.91	10.23	11.50
	Soln. Time (CPU sec.)	**4484.05**	4585.46	4112.12	3644.57	4316.74	4465.80	5293.47	4517.57	4321.70	4481.49	5101.58
	E[W] per period (min)	**12.62**	12.62	12.39	12.38	12.05	12.68	13.24	12.83	12.50	11.98	13.51
$\frac{co}{cw} = 1$	Avg. Patient Waiting (min)	**3.79**	3.79	3.72	3.71	3.61	3.80	3.97	3.85	3.75	3.59	4.05
	E[I] per period (min)	**14.61**	14.71	14.51	14.53	14.72	15.04	14.65	14.24	14.71	14.56	14.42
	E[O] per period (min)	**16.30**	15.90	16.74	16.60	15.45	16.45	17.23	16.74	16.01	15.10	16.76

Session length of each period (dt) is 82.55 overtime are equally important, average patient time decreases drastically however OR utilization is lowered to 82.3%. When solution times are compared, $\frac{co}{cw} = 33$ is the shortest and $\frac{co}{cw} = 10$ is the longest. It is harder to find the optimal solution when a certain cost is not dominant over the other one.

5.3.2 Comparison of Exact and Heuristic Solution Methods

To compare the performance of the heuristic solution methods with the exact ones, we solve the problem by both HDM heuristic and GA. The results for 10 instances are given in Table 5.16

As the importance of overtime decreases, HDM heuristic performs worse. In HDM, first problem considers only the overtime cost which means as waiting time gets more important, it ignores more while assigning periods to the surgeries. In the GA, the performance are similar except $\frac{co}{cw} = 1$. It uses VarA/p25 for calculating the fitness function. These rules perform well in establishing a good solution for sequencing and scheduling, however as the waiting time cost is equal to overtime cost, it gets weaker in mimicking the optimal sequence and schedule.

When HDM and GA are compared in terms of the solution quality and the computational performance, HDM finds a solution with maximum of 1.71% optimality gap, and almost all solutions has optimality gap lower than 1%. However, it takes around 20% more computational time than GA. GA is faster and the solution quality is still great. The worst optimality gap for GA is 3% and in the settings except $\frac{co}{cw} = 1$, the results are really similar to HDM.

To observe the performance of exact and heuristic methods in larger instances, we conducted our experiments on 15 surgeries. Again, we used the surgery mix from "Pain Medicine" surgical group of OPC given in [7] and the problem setting is given in Table 5.17. Due to number of variables and constraints increasing exponentially by increasing the number of surgeries, to obtain solutions in reasonable times, $|S| = 100$ is chosen. As the solution time limit, 9000 CPU seconds is used.

The results of for different $\frac{co}{cw}$ ratios are given in Table 5.18. For calculating the optimality gap for heuristic methods, best lower bound found in exact solution is used.

Table 5.16: Comparison of heuristic solution methods with exact solution

		A-BDA-Full	Hierarchical Decision Making Heu.				Genetic Algorithm			
		Soln. Time	% Opt. Gap	Soln. Time			% Opt. Gap			Soln. Time
				Total	HLP	LLP	Best	Average	Worst	
$\frac{co}{cw} = 33$	Ins 1	4229.79	0.00%	94.16	62.05	32.11	0.00%	0.02%	0.08%	78.08
	Ins 2	4328.55	0.00%	83.68	52.08	31.60	0.00%	0.00%	0.00%	73.97
	Ins 3	3654.75	0.00%	114.05	83.46	30.59	0.26%	0.26%	0.26%	74.85
	Ins 4	4253.67	0.00%	84.99	56.43	28.56	0.00%	0.30%	0.99%	75.83
	Ins 5	4112.03	0.02%	91.52	54.52	37.00	0.00%	0.00%	0.00%	76.62
	Ins 6	3770.25	0.12%	96.12	62.51	33.61	0.00%	0.03%	0.19%	76.30
	Ins 7	4173.47	0.00%	93.36	62.16	31.20	0.00%	0.00%	0.00%	66.69
	Ins 8	4508.32	0.00%	88.69	54.89	33.80	0.00%	0.07%	0.40%	73.51
	Ins 9	4141.48	0.00%	93.17	57.91	35.26	0.00%	0.06%	0.31%	72.42
	Ins 10	4067.39	0.00%	85.35	56.80	28.55	0.00%	0.14%	0.78%	71.11
	Average	**4123.97**	**0.01%**	**92.51**	**60.28**	**32.23**	**0.03%**	**0.09%**	**0.30%**	**73.94**
$\frac{co}{cw} = 10$	Ins 1	11645.00	0.00%	101.88	63.87	38.01	0.00%	0.06%	0.09%	86.88
	Ins 2	9934.74	0.00%	90.91	54.46	36.45	0.00%	0.15%	0.66%	79.54
	Ins 3	9304.76	0.08%	83.66	48.49	35.17	0.00%	0.31%	0.38%	73.56
	Ins 4	12112.10	0.00%	87.75	50.82	36.93	0.00%	0.04%	0.05%	80.74
	Ins 5	8978.17	0.33%	103.59	64.34	39.25	0.00%	0.09%	0.51%	82.48
	Ins 6	9654.63	0.06%	97.83	62.54	35.29	0.00%	0.01%	0.06%	89.97
	Ins 7	11205.10	0.00%	102.64	68.00	34.64	0.00%	0.03%	0.29%	78.75
	Ins 8	12320.80	0.18%	95.29	55.21	40.08	0.17%	0.19%	0.24%	82.41
	Ins 9	12468.60	0.00%	86.53	54.20	32.33	0.00%	0.02%	0.11%	75.88
	Ins 10	13259.20	0.00%	114.17	79.78	34.39	0.00%	0.02%	0.17%	77.67
	Average	**11088.31**	**0.06%**	**96.43**	**60.17**	**36.25**	**0.02%**	**0.09%**	**0.26%**	**80.79**
$\frac{co}{cw} = 1$	Ins 1	4585.46	0.32%	100.79	58.60	42.19	1.79%	1.93%	2.49%	76.43
	Ins 2	4112.12	0.03%	83.86	48.39	35.47	0.00%	0.09%	0.75%	75.30
	Ins 3	3644.57	1.71%	86.86	49.26	37.60	2.52%	2.58%	3.08%	70.72
	Ins 4	4316.74	0.67%	91.93	51.01	40.92	3.38%	3.39%	3.43%	72.03
	Ins 5	4465.80	1.05%	92.73	47.72	45.01	2.58%	2.60%	2.60%	79.47
	Ins 6	5293.47	0.22%	94.99	57.77	37.22	0.20%	0.21%	0.22%	80.39
	Ins 7	4517.57	0.48%	90.97	59.70	31.27	0.30%	0.45%	1.10%	79.62
	Ins 8	4321.70	1.57%	101.47	59.40	42.07	2.69%	2.78%	3.58%	78.75
	Ins 9	4481.49	0.69%	89.56	54.19	35.37	2.75%	2.80%	2.81%	77.64
	Ins 10	5101.58	0.07%	88.23	47.10	41.13	2.81%	2.81%	2.81%	84.59
	Average	**4484.05**	**0.68%**	**92.14**	**53.31**	**38.83**	**1.90%**	**1.96%**	**2.29%**	**77.49**

Solution times are in CPU seconds.

Table 5.17: Experiment setting of larger instances

$\|N\|$	15
$(\mu_{Type1}, \sigma_{Type1}), \|N_{Type1}\|$	$(20.49, 10.86), 2$
$(\mu_{Type2}, \sigma_{Type2}), \|N_{Type2}\|$	$(20.93, 15.08), 8$
$(\mu_{Type3}, \sigma_{Type3}), \|N_{Type3}\|$	$(34.01, 17.42), 5$
$\|T\|$	3
dt	126.1
cw	1
cs	0.01
cu	0

For GA, we made 10 replication for each instance. For all instances, A-BDA-Full failed to find the optimal solution in the time limit. We observe from the optimality gaps that HDM performs slightly better than GA in terms of solution quality. However, GA outperforms HDM when solution time is considered. For these instances, instead of running A-BDA-Full for 9000 CPU seconds, we can reach similar quality solutions by using HDM or GA in much shorter times.

5.3.3 Expected Value of Perfect Information and Value of Stochastic Solution

By using stochastic programming approach, we try to better represent real-life practices through considering uncertainties. To understand the significance of uncertainties and the importance of future information, VSS and EVPI are calculated by using the following equations.

$$EVPI = \frac{RP - WS}{RP}$$
$$VSS = \frac{EEV - RP}{EEV}$$

In some instances, optimal solution may not be reached within the time limit. For these instances, VSS calculated by the best solution is a lower bound on the accurate values of VSS. Meanwhile, EVPI calculated by the best solution found is an upper bound on the accurate values of EVPI. To observe the effect of $\frac{co}{cw}$ on considering

Table 5.18: Comparison of exact and heuristic solution methods on larger instances

		A-BDA-Full		HDM		Genetic Algorithm			
		% Opt. Gap	Soln. Time	% Opt. Gap	Soln. Time	Best % Opt. Gap	Avg. % Opt. Gap	Worst % Opt. Gap	Soln. Time
$\frac{co}{cw} = 33$	Ins 1	36.33%	9000.00	35.97%	292.05	36.16%	37.06%	37.93%	658.62
	Ins 2	42.05%	9000.00	41.83%	640.65	41.83%	43.52%	45.72%	609.76
	Ins 3	48.50%	9000.00	47.69%	257.75	48.21%	49.06%	50.33%	357.79
	Ins 4	46.71%	9000.00	46.15%	1786.75	46.54%	47.37%	49.04%	424.73
	Ins 5	42.30%	9000.00	41.73%	1127.66	42.89%	43.44%	44.17%	725.71
	Ins 6	43.89%	9000.00	43.46%	377.95	43.28%	43.91%	44.54%	538.25
	Ins 7	42.95%	9000.00	42.26%	993.71	42.26%	43.06%	43.98%	579.26
	Ins 8	44.17%	9000.00	43.69%	313.61	43.85%	44.68%	45.74%	350.94
	Ins 9	41.52%	9000.00	40.52%	284.39	40.91%	41.83%	43.39%	408.32
	Ins 10	46.78%	9000.00	45.73%	1089.31	45.91%	46.76%	47.61%	506.97
	Average	**43.52%**	**9000.00**	**42.90%**	**716.38**	**43.18%**	**44.07%**	**45.24%**	**516.04**
$\frac{co}{cw} = 10$	Ins 1	47.47%	9000.00	46.47%	740.03	46.48%	47.01%	48.01%	893.91
	Ins 2	53.52%	9000.00	52.44%	878.73	52.38%	53.51%	55.01%	878.01
	Ins 3	58.99%	9000.00	58.34%	702.04	58.59%	58.92%	59.60%	701.51
	Ins 4	57.30%	9000.00	56.64%	3156.53	56.10%	56.85%	57.75%	811.99
	Ins 5	53.66%	9000.00	53.23%	1323.59	53.18%	53.65%	54.29%	727.42
	Ins 6	55.13%	9000.00	54.63%	953.36	54.67%	55.22%	55.82%	1054.46
	Ins 7	52.91%	9000.00	52.55%	1185.83	52.73%	53.51%	54.68%	752.15
	Ins 8	55.47%	9000.00	54.81%	970.63	54.50%	55.50%	56.17%	747.37
	Ins 9	52.88%	9000.00	52.11%	541.79	52.19%	52.92%	53.70%	561.90
	Ins 10	56.48%	9000.00	55.79%	1456.78	55.86%	56.32%	56.97%	791.09
	Average	**54.38%**	**9000.00**	**53.70%**	**1190.93**	**53.67%**	**54.34%**	**55.20%**	**791.98**
$\frac{co}{cw} = 1$	Ins 1	64.72%	9000.00	64.87%	265.39	64.30%	64.74%	65.27%	458.89
	Ins 2	72.10%	9000.00	71.51%	296.45	72.18%	72.61%	73.15%	686.20
	Ins 3	74.22%	9000.00	73.81%	264.34	73.87%	74.37%	74.76%	313.36
	Ins 4	71.51%	9000.00	71.64%	908.10	71.38%	71.90%	72.31%	664.79
	Ins 5	70.13%	9000.00	70.46%	547.95	70.18%	70.42%	71.21%	394.47
	Ins 6	72.95%	9000.00	72.42%	379.55	72.36%	72.64%	72.87%	567.17
	Ins 7	69.38%	9000.00	69.89%	599.26	69.72%	70.02%	70.39%	296.32
	Ins 8	73.55%	9000.00	73.12%	605.62	72.82%	73.35%	74.05%	547.83
	Ins 9	69.62%	9000.00	70.02%	305.53	69.72%	70.19%	70.53%	426.21
	Ins 10	73.98%	9000.00	74.68%	854.18	74.07%	74.38%	74.68%	417.44
	Average	**71.22%**	**9000.00**	**71.24%**	**502.64**	**71.06%**	**71.46%**	**71.92%**	**477.27**

Solution times are in CPU seconds.

uncertainty and value of information, VSS and EVPI are calculated for each setting and the results are represented in Table 5.19.

Table 5.19: VSS and EVPI of each instance for three different overtime cost parameter values

	$\frac{co}{cw} = 1$		$\frac{co}{cw} = 10$		$\frac{co}{cw} = 33$	
	EVPI	VSS	EVPI	VSS	EVPI	VSS
Instance 1	79.5%	12.1%	56.4%	12.1%	48.3%	18.7%
Instance 2	78.2%	16.3%	55.0%	13.4%	46.9%	19.9%
Instance 3	77.7%	16.6%	54.2%	12.1%	46.2%	19.2%
Instance 4	79.7%	15.6%	57.3%	11.7%	49.1%	20.2%
Instance 5	78.6%	10.9%	55.1%	12.9%	47.1%	20.4%
Instance 6	77.0%	7.5%	51.9%	11.7%	43.4%	18.9%
Instance 7	77.8%	10.2%	54.0%	10.8%	45.8%	17.5%
Instance 8	79.1%	13.7%	56.0%	12.3%	47.9%	19.9%
Instance 9	80.6%	11.9%	58.8%	11.2%	50.9%	20.4%
Instance 10	79.0%	9.2%	55.3%	12.1%	47.2%	19.2%
Average	**78.7%**	**12.4%**	**55.4%**	**12.0%**	**47.3%**	**19.4%**

As the $\frac{co}{cw}$ ratio decreases, EVPI increases and VSS decreases. In the case of higher overtime cost, it is more important to plan considering uncertainties because wrong time allocations will cost more. This results in increase of VSS. However, in lower $\frac{co}{cw}$, planning and scheduling decisions get more important since any patient waiting is equally important as overtime. Knowing more about the uncertainties will bring more benefit.

CHAPTER 6

CONCLUSION

In this study, we worked on multi-period appointment planning and scheduling problem (APSP) under surgery time uncertainty. The problem includes the assignment of surgeries to a period in the planning horizon, sequencing of the surgeries in each period and allocating time for each surgery. All these decisions are made simultaneously. The problem is modeled as a two-stage stochastic program where the first stage includes all planning and scheduling decisions. In the second stage, waiting time of patients, and idle time and overtime of the OR is calculated. The objective function is composed of expected costs of these waiting time, idle time and overtime values.

APSP is a NP-hard problem and there are many identical solutions since the periods are identical. For eliminating these identical solutions and enhancing our model formulation, we used symmetry breaking constraints. Main idea of these constraints is that each period should have less surgery than the previous period.

To solve our two-stage stochastic programming model, we used extensive formulation and decomposition methods such as the L-shaped method with single and multi-cut variations, L-shaped method in branch-and-cut framework and Benders' decomposition. In our preliminary experiments, we worked on small instances with eight surgeries to be scheduled in three periods. We used up to 200 scenarios. We observed that solving the extensive formulation performs better than the classic single-cut L-shaped method for small instances. In addition, MVC, cuts based on mean value scenario, are considered to improve the computational performance of L-shaped algorithm.

In our problem, it is important to carry significant level of information through optimality cuts. All multi-cut L-shaped methods work better than the single-cut L-shaped methods; and decomposing subproblems further and generating more cuts even works better. This is valid for both iterative and branch-and-cut frameworks. Also, working on the same branch-and-bound tree rather than creating and exploring a new one in each iteration performs better, and hence B&C framework decreases the solution time by a great amount.

Besides the L-shaped method, we also used Benders' decomposition algorithm which performed best in terms of computational performance. Considering only period assignment and sequencing variables in the master problem and moving time allocation decisions to subproblems performs well. Since subproblems are still LP, they are solved easily, and cuts generated from these subproblems are tighter. However, this observation only holds for the automatic Benders' decomposition algorithm, embedded in CPLEX. When it is coded through only B&C framework, it performs worse than both mutli-cut L-shaped and multi-cut L-shaped with subproblems decomposed based on both scenarios and periods. We believe that this is due to the other type of cuts used by CPLEX in the automatic BDA.

By using simple sequencing rules, we analyzed the dynamics behind each decision in our problem. When the cost of overtime is significantly higher than patient waiting time and idle time, LPT, VarD and CoefD work well at assigning surgeries to periods and SPT, VarA and CoefA work well in sequencing. Since our surgery times are generated from a log-normal distribution, which is right-skewed, using 25^{th} percentile as job hedging level performed well.

We developed two heuristic algorithms, hierarchical decision making heuristic and genetic algorithm. In hierarchical decision making heuristic, as a first step an easy model is solved just for placing surgeries to periods, considering only overtime costs. In the second step, period assignments are fixed and the remaining problem is solved to optimality. Similarly, GA tries to find the best period assignments and evaluate the quality of solutions through best working simple sequencing and job hedging rules (-/VarA/p25). To find the final schedule, sequencing and scheduling problem is solved to optimality, as in the second step of the HDM.

In our main experiments, we analyzed the effect of cost parameters on our solutions. We solved three different cases of cost parameter ratios in larger instances of ten surgeries with 800 scenarios. Idle time and overtime increases as cost of overtime decreases. Also, it is easier to find the optimal solution when one cost is significantly higher than the other.

Then, we compared the solutions returned by our heuristics with the optimal solution. Both heuristics perform well within reasonable computational times. We observe that solving the problem hierarchically is good enough and computationally much more efficient than solving the integrated problem in 20 times more time, even in the case with $\frac{co}{cw} = 1$. Genetic algorithm also performs well and find near-optimal solutions; however, HDM obtains closer solutions to the optimal. Since it is really fast, it is promising to use GA in much larger instances and it will find "good solutions" in still reasonable time. In the larger instances, it is not possible to reach optimal solution by exact methods. We can reach a solution with similar quality by using heuristic methods instead of running the exact methods for very long times. Also, solution performances of HDM and GA are close to each other in terms of solution quality. GA reaches a similar solution with a better computational performance.

Also, when simple sequencing rules are considered for period assignment, they do not perform well and the best performing rules result in optimality gap of around 8%. When results of experiments for HDM is considered, solving an easy problem, which only considers the overtime cost for surgery-to-period assignments, performs much better rather than using these simple rules.

We also calculated EVPI and VSS for three different ratios of cost of overtime and waiting time. We observed that VSS increases and EVPI decreases as this ratio increases. When overtime is more important, including uncertainty in the model becomes more important.

For the future research directions, lower bounds can be found for the objective function value to better evaluate the performance of the heuristic methods. In GA, different methods for creating initial population and/or different crossover probabilities and/or different mutation probabilities can be tested.

REFERENCES

[1] P. M. V. Bosch, D. C. Dietz, and J. R. Simeoni, "Scheduling customer arrivals to a stochastic service system," *Naval Research Logistics (NRL)*, vol. 46, pp. 549–559, 8 1999.

[2] OECD, *Health at a Glance 2017.* 2017.

[3] A. Rais and A. Viana, "Operations research in healthcare: a survey," *International Transactions in Operational Research*, vol. 18, no. 1, pp. 1–31, 2011.

[4] K. A. Cullen, M. J. Hall, and A. Golosinskiy, "Ambulatory surgery in the United States, 2006.," *National health statistics reports*, 2009.

[5] B. Berg and B. T. Denton, "Appointment planning and scheduling in outpatient procedure centers," in *Handbook of Healthcare System Scheduling* (R. Hall, ed.), pp. 131–154, Boston, MA: Springer US, 2012.

[6] B. Berg, B. Denton, H. Nelson, H. Balasubramanian, A. Rahman, A. Bailey, and K. Lindor, "A discrete event simulation model to evaluate operational performance of a colonoscopy suite," *Medical Decision Making*, vol. 30, no. 3, pp. 380–387, 2010. PMID: 19773583.

[7] S. Gul, B. T. Denton, J. W. Fowler, and T. Huschka, "Bi-criteria scheduling of surgical services for an outpatient procedure center," *Production and Operations Management*, vol. 20, no. 3, pp. 406–417, 2011.

[8] J. H. May, W. E. Spangler, D. P. Strum, and L. G. Vargas, "The surgical scheduling problem: Current research and future opportunities," *Production and Operations Management*, vol. 20, no. 3, pp. 392–405, 2011.

[9] F. Sabria and C. F. Daganzo, "Approximate expressions for queueing systems with scheduled arrivals and established service order," *Transportation Science*, vol. 23, no. 3, pp. 159–165, 1989.

[10] C. Guan and R. R. Liu, "Container terminal gate appointment system optimization," *Maritime Economics & Logistics*, vol. 11, pp. 378–398, Dec 2009.

[11] I. Bendavid and B. Golany, "Setting gates for activities in the stochastic project scheduling problem through the cross entropy methodology," *Annals of Operations Research*, vol. 172, p. 259, May 2009.

[12] J. Castaing, A. Cohn, B. T. Denton, and A. Weizer, "A stochastic programming approach to reduce patient wait times and overtime in an outpatient infusion center," *IIE Transactions on Healthcare Systems Engineering*, vol. 6, no. 3, pp. 111–125, 2016.

[13] D. Gupta and B. Denton, "Appointment scheduling in health care: Challenges and opportunities," *IIE Transactions*, vol. 40, no. 9, pp. 800–819, 2008.

[14] S. Zhu, W. Fan, S. Yang, J. Pei, and P. M. Pardalos, "Operating room planning and surgical case scheduling: a review of literature," *Journal of Combinatorial Optimization*, vol. 37, pp. 757–805, Apr 2019.

[15] T. Cayirli and E. Veral, "Outpatient scheduling in health care: a review of literature," *Production and Operations Management*, vol. 12, no. 4, pp. 519–549, 2003.

[16] A. Ahmadi-Javid, Z. Jalali, and K. J. Klassen, "Outpatient appointment systems in healthcare: A review of optimization studies," *European Journal of Operational Research*, vol. 258, no. 1, pp. 3 – 34, 2017.

[17] N. T. J. Bailey, "A study of queues and appointment systems in hospital outpatient departments, with special reference to waiting-times," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 14, no. 2, pp. 185–199, 1952.

[18] M. Samudra, C. Van Riet, E. Demeulemeester, B. Cardoen, N. Vansteenkiste, and F. E. Rademakers, "Scheduling operating rooms: achievements, challenges and pitfalls," *Journal of Scheduling*, vol. 19, pp. 493–525, Oct 2016.

[19] B. Cardoen, E. Demeulemeester, and J. BeliÃ«n, "Operating room planning and scheduling: A literature review," *European Journal of Operational Research*, vol. 201, no. 3, pp. 921 – 932, 2010.

[20] B. Zeng, A. Turkcan, J. Lin, and M. Lawley, "Clinic scheduling models with overbooking for patients with heterogeneous no-show probabilities," *Annals of Operations Research*, vol. 178, pp. 121–144, Jul 2010.

[21] L. R. LaGanga and S. R. Lawrence, "Appointment overbooking in health care clinics to improve patient service and clinic performance," *Production and Operations Management*, vol. 21, no. 5, pp. 874–888, 2012.

[22] S. A. Erdogan and B. Denton, "Dynamic appointment scheduling of a stochastic server with uncertain demand," *INFORMS Journal on Computing*, vol. 25, no. 1, pp. 116–132, 2013.

[23] Y. Gocgun and M. L. Puterman, "Dynamic scheduling with due dates and time windows: an application to chemotherapy patient appointment booking," *Health Care Management Science*, vol. 17, pp. 60–76, Mar 2014.

[24] N. Liu, S. Ziya, and V. G. Kulkarni, "Dynamic scheduling of outpatient appointments under patient no-shows and cancellations," *Manufacturing & Service Operations Management*, vol. 12, no. 2, pp. 347–364, 2010.

[25] L. W. Robinson and R. R. Chen, "Scheduling doctors' appointments: optimal and empirically-based heuristic policies," *IIE Transactions*, vol. 35, no. 3, pp. 295–307, 2003.

[26] A. Saremi, P. Jula, T. ElMekkawy, and G. G. Wang, "Appointment scheduling of outpatient surgical services in a multistage operating room department," *International Journal of Production Economics*, vol. 141, no. 2, pp. 646 – 658, 2013. Special Issue on Service Science.

[27] B. Denton and D. Gupta, "A sequential bounding approach for optimal appointment scheduling," *IIE Transactions*, vol. 35, no. 11, pp. 1003–1016, 2003.

[28] B. Denton, J. Viapiano, and A. Vogl, "Optimization of surgery sequencing and scheduling decisions under uncertainty," *Health Care Management Science*, vol. 10, pp. 13–24, Feb 2007.

[29] C. Mancilla and R. Storer, "A sample average approximation approach to stochastic appointment sequencing and scheduling," *IIE Transactions*, vol. 44, no. 8, pp. 655–670, 2012.

[30] B. P. Berg, B. T. Denton, S. A. Erdogan, T. Rohleder, and T. Huschka, "Optimal booking and scheduling in outpatient procedure centers," *Computers & Operations Research*, vol. 50, pp. 24 – 37, 2014.

[31] S. Batun, B. T. Denton, T. R. Huschka, and A. J. Schaefer, "Operating room pooling and parallel surgery processing under uncertainty," *INFORMS Journal on Computing*, vol. 23, no. 2, pp. 220–237, 2011.

[32] K. S. Shehadeh, A. E. Cohn, and M. A. Epelman, "Analysis of models for the stochastic outpatient procedure scheduling problem," *European Journal of Operational Research*, vol. 279, no. 3, pp. 721 – 731, 2019.

[33] S. A. Erdogan and B. T. Denton, *Surgery Planning and Scheduling*. American Cancer Society, 2011.

[34] B. Pang, X. Xie, Y. Song, and L. Luo, "Surgery scheduling under case cancellation and surgery duration uncertainty," *IEEE Transactions on Automation Science and Engineering*, vol. 16, pp. 74–86, Jan 2019.

[35] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming. [electronic resource].* Springer Series in Operations Research and Financial Engineering, Springer New York, 2011.

[36] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische mathematik*, vol. 4, no. 1, pp. 238–252, 1962.

[37] A. Madansky, "Inequalities for stochastic linear programming problems," *Management Science*, vol. 6, no. 2, pp. 197–204, 1960.

[38] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, pp. 82 – 117, 2013. Prediction, Control and Diagnosis using Advanced Neural Computations.

[39] J. H. Holland *et al.*, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[40] B. T. Denton, A. J. Miller, H. J. Balasubramanian, and T. R. Huschka, "Optimal

allocation of surgery blocks to operating rooms under uncertainty," *Operations Research*, vol. 58, no. 4-part-1, pp. 802–816, 2010.

APPENDIX A

UPPER AND LOWER BOUNDS ON x_{kt}

<u>Upper Bound</u>

Let ρ_l: l^{th} value when $(d_i^{max} - d_i^{min})$ for each $i \in N^r$ are sorted in decreasing order.

<u>For a day t and for $k = 1$:</u>

$$\sum_{i \in N^r} d_i^{min} \cdot y_{i1t} \leq x_{1t} \leq \sum_{i \in N^r} d_i^{max} \cdot y_{i1t}$$

<u>For $k = 2$:</u>

$$\sum_{i \in N^r} d_i^{min} \cdot y_{i1t} + \sum_{i \in N^r} d_i^{min} \cdot y_{i2t} \leq x_{1t} + x_{2t} \leq \sum_{i \in N^r} d_i^{max} \cdot y_{i1t} + \sum_{i \in N^r} d_i^{max} \cdot y_{i2t}$$

$$-\sum_{i \in N^r} d_i^{max} \cdot y_{i1t} \leq -x_{1t} \leq -\sum_{i \in N^r} d_i^{min} \cdot y_{i1t}$$

$$x_{1t} - x_{1t} + x_{2t} \leq \sum_{i \in N^r} d_i^{max} \cdot y_{i1t} - \sum_{i \in N^r} d_i^{min} \cdot y_{i1t} + \sum_{i \in N^r} d_i^{max} \cdot y_{i2t}$$

$$x_{2t} \leq \sum_{i \in N^r} \left(d_i^{max} - d_i^{min}\right) \cdot y_{i1t} + \sum_{i \in N^r} d_i^{max} \cdot y_{i2t}$$

$$x_{2t} \leq \max_i \left(d_i^{max} - d_i^{min}\right) \cdot y_{i1t} + \sum_{i \in N^r} d_i^{max} \cdot y_{i2t}$$

$$x_{2t} \leq \sum_{i \in N^r} \left(\rho_1 + d_i^{max}\right) \cdot y_{i2t}$$

$$x_{k't} \leq \sum_{i \in N^r} \left(\sum_{l=1}^{k'-1} \left(\rho_l + d_i^{max}\right) \right) \cdot y_{ik't}$$

Lower Bound

Let j be the 2^{nd} and l be the 3^{rd} surgery on period t

For $k = 2$;

$$P_{l3ts} - S_{l3ts} = P_{j2ts} + d_{js} \cdot y_{j2t} - x_{2t} \quad \forall s$$

$$d_{js} \cdot y_{j2t} - x_{2t} \le P_{l3ts} - S_{l3ts} \le \left(d_i^{max} - d_i^{min}\right) + d_{js} \cdot y_{j2t} - x_{2t}$$

Let $x_{2t} = d_j^{min}$

$$\begin{aligned}
d_j^s \cdot y_{j2t} - d_j^{min} \le & P_{l3t}^s - S_{l3t}^s && \le \left(d_i^{max} - d_i^{min}\right) + d_j^s \cdot y_{j2t} - d_j^{min} \\
d_j^{min} - d_j^{min} \le & \quad P_{l3t}^s && \\
& \quad P_{l3t}^s && \le \left(d_i^{max} - d_i^{min}\right) + (d_j^{max} - d_j^{min}) \\
0 \le & \quad P_{l3t}^s && \le \left(d_i^{max} - d_i^{min}\right) + (d_j^{max} - d_j^{min})
\end{aligned}$$

$LHS_1 = 0$ and $RHS_1 = (d_i^{max} - d_i^{min}) + (d_j^{max} - d_j^{min})$

Let $x_{2t} = d_j'^{min}$ where $d_j'^{min} < d_j^{min}$

$$\begin{aligned}
d_j^s \cdot y_{j2t} - d_j'^{min} \le & P_{l3t}'^s - S_{l3t}'^s && \le \left(d_i^{max} - d_i^{min}\right) + d_j^s \cdot y_{j2t} - d_j'^{min} \\
\left(d_j^{min} - d_j'^{min}\right) \le & \quad P_{l3t}'^s && \\
& \quad P_{l3t}'^s && \le \left(d_i^{max} - d_i^{min}\right) + (d_j^{max} - d_j'^{min}) \\
\left(d_j^{min} - d_j'^{min}\right) \le & \quad P_{l3t}'^s && \le \left(d_i^{max} - d_i^{min}\right) + (d_j^{max} - d_j'^{min})
\end{aligned}$$

$LHS_2 = (d_j^{min} - d_j'^{min}) > 0$ and $RHS_2 = (d_i^{max} - d_i^{min}) + (d_j^{max} - d_j'^{min})$

$|LHS_1| < |LHS_2|$ and $|RHS_1| < |RHS_2|$ so $P_{l3ts} < P'_{l3ts}$ $\quad \square$

We can extend and generalize this lower bounds for x_{kt} where $k \ge 2$ such as:

$$\sum_{i=1}^{N} d_i^{min} y_{ikt} \le x_{kt} \qquad k \in K, t \in T$$

APPENDIX B

UPPER BOUND ON P^s_{ikt}

For $k = 1$

$$P^s_{j2t} - S^s_{j2t} = d^s_i * y_{i1t} - x_{1t}$$

$$P^s_{j2t} \leq \left(d^s_i - d^{min}_i\right) y_{i1t}$$

$$\leq (\max_i d^s_i - \min_i d^{min}_i) y_{j2t}$$

For the indices to match, we need the larger upper bound. If no job is assigned to a position, there cannot be any waiting time.

For $k = 2$

$$P^s_{l3t} - S^s_{l3t} = P^s_{j2t} + d^s_j * y_{j2t} - x_{2t}$$

$$P^s_{l3t} \leq (\max_i d^s_i - \min_i d^{min}_i) y_{l3t} + \max_i d^s_i * y_{l3t} - \min_i d^{min}_i * y_{l3t}$$

Since we used max and min values for one position, and a patient cannot be assigned into two positions, we can use the 2nd max and 2nd min values for the upper bound.

$$P^s_{l3t} \leq (\max_i d^s_i - \min_i d^{min}_i) y_{l3t} + \max_i d^s_i * y_{l3t} - \min_i d^{min}_i * y_{l3t}$$

$$P^s_{l3t} \leq \left(\sum_{z=1}^{2} \left(dsc^s_z - d^{mins}_z\right)\right) * y_{l3t}$$

dsc^s_k: k^{th} longest surgery duration in scenario s and

d^{smin}_k: k^{th} shortest surgery duration after sorting d^{min}_i in increasing order.

We can extend and generalize this bound such as:

$$P^s_{ikt} \leq \left(\sum_{l=1}^{k-1} \left(dsc^s_l - d^{mins}_l\right)\right) * y_{ikt}$$

APPENDIX C

GA RESULTS

Table C.1: GA results for Initial Population Size=30

PopSize	XO	MUT	PS	NEXTGEN	Opt Gap	# of Times Opt. Soln. Found	Best Opt. Gap from Init. Pop.	Solution Time (CPU sec.)			# Total Iter	# of Best Soln. Changes	Opt. Gap of Soln w/ $-/VarA/p25$	Best Opt. Gap from Init Pop. Soln. w/ $-/VarA/p25$
								Total	For Optimization	For Iterations				
30	Uniform Crossover	M1	Random	wPbO	0.46%	3.3	5.44%	7.79	7.63	0.16	2130	2.62	1.93%	6.97%
				2of4	0.06%	6.9	6.74%	7.81	7.61	0.20	2607	4.12	1.45%	8.23%
				worst2	0.36%	4.5	6.88%	7.65	7.47	0.18	2308	2.63	1.86%	8.44%
			Fitness	wPbO	0.62%	3.5	5.65%	7.72	7.55	0.16	2118	2.45	2.06%	7.14%
				2of4	0.06%	7.0	6.01%	7.59	7.39	0.20	2574	3.58	1.45%	7.46%
				worst2	0.60%	4.4	5.64%	7.76	7.57	0.19	2399	2.74	2.06%	7.07%
		M2	Random	wPbO	0.45%	3.5	6.10%	7.74	7.58	0.16	2108	2.53	1.90%	7.59%
				2of4	0.07%	6.8	6.52%	7.61	7.42	0.19	2494	3.87	1.46%	8.03%
				worst2	0.53%	4.6	6.40%	7.67	7.49	0.18	2336	3.05	1.99%	7.82%
			Fitness	wPbO	0.53%	3.0	6.33%	7.73	7.57	0.16	2117	2.56	2.06%	7.83%
				2of4	0.09%	6.8	6.05%	7.55	7.35	0.20	2517	3.79	1.48%	7.50%
				worst2	0.45%	4.4	7.17%	7.87	7.69	0.19	2340	2.85	1.91%	8.68%
	1-Point Crossover	M1	Random	wPbO	1.39%	2.1	6.37%	7.88	7.72	0.16	2129	2.42	2.88%	7.85%
				2of4	1.29%	2.4	6.81%	7.75	7.57	0.18	2350	2.78	2.76%	8.31%
				worst2	2.69%	1.1	6.50%	7.63	7.47	0.17	2163	1.73	4.24%	8.01%
			Fitness	wPbO	1.37%	2.1	6.21%	7.83	7.67	0.16	2105	2.07	2.88%	7.67%
				2of4	1.24%	2.7	6.16%	7.82	7.64	0.18	2300	2.10	2.73%	7.63%
				worst2	2.10%	1.8	5.84%	7.80	7.63	0.16	2101	1.68	3.60%	7.34%
		M2	Random	wPbO	1.46%	1.5	5.93%	7.64	7.48	0.16	2109	2.18	2.99%	7.39%
				2of4	0.98%	2.4	6.58%	7.58	7.40	0.18	2382	2.82	2.44%	8.06%
				worst2	2.51%	1.1	6.12%	7.67	7.50	0.17	2248	1.91	4.00%	7.59%
			Fitness	wPbO	1.39%	1.3	6.31%	7.69	7.53	0.16	2109	2.18	2.90%	7.81%
				2of4	1.30%	2.6	6.66%	7.74	7.56	0.18	2282	2.19	2.79%	8.17%
				worst2	2.16%	1.1	6.45%	7.76	7.60	0.16	2113	1.77	3.69%	7.91%

Table C.2: GA results for Initial Population Size=100

PopSize	XO	MUT	PS	NEXTGEN	Opt Gap	# of Times Opt. Soln. Found	Best Opt. Gap from Init. Pop.	Solution Time (CPU sec.) Total	For Optimization	For Iterations	# Total Iter	# of Best Soln. Changes	Opt. Gap of Soln w/ $-/VarA/p25$	Best Opt. Gap from Init Pop. Soln. w/ $-/VarA/p25$
100	Uniform Crossover	M1	Random	wPbO	0.05%	7.1	3.08%	7.60	7.41	0.19	2476	2.63	1.45%	4.58%
				2of4	0.05%	7.0	3.03%	7.80	7.60	0.20	2583	2.77	1.44%	4.48%
				worst2	0.06%	6.8	3.09%	7.62	7.41	0.21	2582	2.56	1.45%	4.59%
			Fitness	wPbO	0.08%	6.6	3.32%	7.92	7.71	0.21	2486	2.69	1.47%	4.81%
				2of4	0.06%	6.9	3.26%	7.84	7.61	0.23	2698	2.82	1.45%	4.74%
				worst2	0.06%	7.0	2.98%	7.69	7.47	0.22	2513	2.43	1.46%	4.52%
		M2	Random	wPbO	0.08%	6.5	3.09%	7.66	7.46	0.19	2429	2.70	1.47%	4.55%
				2of4	0.06%	6.9	3.25%	7.57	7.36	0.21	2637	2.72	1.45%	4.73%
				worst2	0.06%	6.8	3.06%	7.38	7.17	0.21	2578	2.53	1.46%	4.57%
			Fitness	wPbO	0.08%	6.5	3.04%	7.57	7.36	0.21	2459	2.58	1.47%	4.51%
				2of4	0.05%	7.0	2.78%	7.73	7.50	0.23	2748	2.85	1.44%	4.29%
				worst2	0.06%	6.8	3.19%	7.66	7.44	0.22	2573	2.57	1.45%	4.63%
	1-Point Crossover	M1	Random	wPbO	0.17%	5.7	3.66%	7.71	7.51	0.20	2596	2.58	1.60%	5.17%
				2of4	0.11%	6.0	3.14%	7.91	7.70	0.21	2632	2.49	1.57%	4.70%
				worst2	0.70%	3.2	2.03%	7.78	7.60	0.18	2213	1.11	2.18%	3.58%
			Fitness	wPbO	0.21%	4.6	2.81%	7.74	7.53	0.21	2493	2.09	1.63%	4.31%
				2of4	0.24%	4.9	2.99%	7.78	7.57	0.21	2519	2.09	1.68%	4.48%
				worst2	0.66%	3.2	2.66%	7.96	7.77	0.19	2200	1.27	2.16%	4.17%
		M2	Random	wPbO	0.18%	5.0	2.91%	7.66	7.46	0.20	2557	2.06	1.61%	4.40%
				2of4	0.17%	5.3	2.79%	7.69	7.48	0.21	2702	2.33	1.57%	4.31%
				worst2	0.68%	3.3	2.95%	7.71	7.52	0.19	2322	1.50	2.18%	4.44%
			Fitness	wPbO	0.27%	4.4	2.78%	7.83	7.63	0.21	2460	1.81	1.71%	4.27%
				2of4	0.17%	5.5	3.38%	7.77	7.55	0.22	2710	2.42	1.63%	4.87%
				worst2	0.46%	3.7	2.79%	8.00	7.81	0.19	2222	1.52	1.97%	4.33%