UNIVERSITY OF SOUTHAMPTON

Online Scheduling in Hospital Theatre Scheduling

by

Nor Aliza Binti Abd Rahmin

A thesis submitted in partial

fulfillment of the requirements for the degree of

Master of Philosophy

in the

Faculty of Social and Human Sciences

School of Mathematical Sciences

December 2, 2018

Declaration of Authorship

I, Nor Aliza Binti Abd Rahmin , declare that this thesis titled, **Online Scheduling in Hospital Theatre Scheduling** and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: 2/12/2018

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF SOCIAL AND HUMAN SCIENCES
SCHOOL OF MATHEMATICAL SCIENCES

Master of Philosophy

by Nor Aliza Binti Abd Rahmin

Increasing population across all age groups has contributed to the increasing demand for health care especially those that require surgeries, thus putting more pressure on hospitals. The inability to provide adequate and efficient treatment as a result of resource constraints causes patients to wait longer for treatment. Waiting for treatment due to unavailability of an operating theatre can result in both deteriorating health and inconvenience. It is even more frustrating when the scheduled operation is cancelled because some slots is used for emergency patients or the scheduled operations are longer that planned. When such situation occurs, some patients need to be rescheduled.

To resolve this problem, an operating theatre scheduling for emergency and regular patients is considered. We consider the single operating theatre problem across multiple days together with the multiple operating theatres problem on a single day. The aim is to minimise the cost incurred when patients need to be rescheduled as well as ensure minimal delay and rescheduling. We develop a model and design an algorithm to schedule operations for patients, taking into account their urgency. Patients' urgency depends on their respective situation and changes depend on several factors.

Tackling the problem of scheduling single operating theatre, we use a heuristic method to provide a starting solution before applying local search and simulating annealing. The schedule is updated daily to take into account variations from planned durations and the arrival of emergency patients. The rescheduling of patients may be necessary. We consider the priority of patients and ensure that top priority patients be considered first in the scheduling and less important patients can be rescheduled if necessary.

Under the local search technique, we swap every pair of patients if they satisfy the conditions imposed. After the patients are swapped, we check the total cost of the swap and compare it with the current cost. If the new total cost is less than the current cost, the swap will be finalised. We then consider the next patient until all remaining patients in the list are accounted for and we come out with the new list of schedule.

Continuing from that, we utilise simulating annealing technique where we calculate the difference of the total cost (total new cost - total current cost), Δ

between a pair of patients that we plan to swap. With this approach, as opposed to the local search procedure, even when the difference of the total cost is positive, swapping might still take place but only with a certain probability.

Besides single operating theatre, we also consider the scheduling of multiple operation theatres in a single day. Rather than using the algorithm technique, we propose an integer programming model, the Zero-One Programming model and develop an algorithm that utilises the model in scheduling multiple parallel OTs. If a surgery runs longer that expected or an emergency patient arrived into the system, patients can be moved between the available OTs to ensure that surgeries can still be performed; or if the model decides it is better to reschedule therefore the patients will be rescheduled to the next day.

In order to test the efficiency of our models and look at the compatibility of the models with our algorithm, data are generated with different parameters to see if our proposed models have the ability to lower cost as well as prevent delays and rescheduling. Moreover, we check the computational time of our algorithm to ascertain whether it can provide solutions within a short amount of time. Overall, our models show improvement in reducing cost and minimising delay and rescheduling.

Contents

List of Abbreviations

COP	Combinatorial Optimization Problem
OT	Operating Theatre
A&E	Accident and emergency
ZOP	Zero-One Programming Model

List of Figures

List of Tables

List of Algorithms

Acknowledgements

"In the name of Allah, the Most Gracious, the Most Merciful"

There is neither might nor strength except with Allah.

My deepest appreciation goes to my supervisors, Prof. Chris Potts and Dr Marion Penn for their continuous guidance, support and valuable discussions throughout this MPhil journey. I would also like to thank Dr Honora Smith and all the people in CORMSIS for all the valuable insights and ideas.

I am also grateful to the Universiti Putra Malaysia and Malaysian Ministry of Higher Education for providing me the scholarship. Special thanks to my best friend Dr Istrinayagy and Dr Norhashidah and my MSc supervisors Dr Mansor, Prof. Malek and Prof. Fudziah for their support before and during my MPhil study.

This journey would have been dull and uninspiring without my friends especially Nargesh, Mol, Mussota, Arash, Mustakim, Iskandar, Abu Zar, Dr Zurina and Ainus for their help, support and faithful friendship. They have kept my moral and spirit high throughout my studies.

Lastly but most importantly, all this will not have been possible without my family especially my beloved husband Tengku Abdul Mutalib, who has been my shoulder to cry on and my dear children: Qayyim, Muqri, Thaqif and Adelia. Mama loves you! Also special thanks to my dear mother and families in Malaysia for all their prayers and supports.

To my family,
Tengku Abdul Mutalib, Tengku Afiq Qayyim, Tengku Adam Muqri, Tengku Afif Thaqif and Tengku Adelia Maisarra.

Chapter 1

Introduction

1.1 Background

Combinatorial Optimization Problem (COP) is the most general of optimization problems, whereby the goal is to find the best solution on a domain when the domain is a finite number of feasible solutions. Nondeterministic polynomial hard time problems (NP-hard problems) are the most important and interesting of these problems. NP-hard problems refer to the class of decision problem that can be solved in polynomial time in a nondeterministic Turing Machine.

It is very hard to predict the optimum solution in a nondeterministic machine because there are multiple possible outcomes for each data input. However, algorithms can be designed to produce approximate optimal solutions. The usage of algorithms to solve the COP is more efficient and faster as discussed by Burke and Kendall (2005), Grotschel and Lovász (1995) and Selman et al. (1992).

The application of COP are found in a variety of areas including the planning and management of operations and resources, investment planning, production scheduling, transportation planning, communications network and health care

scheduling (see Yu (2013), Paschos (2013)). In recent years, health care system is focused on a more efficient health care scheduling to provide high quality services with minimal possible costs .

Health care providers are facing more complicated challenges to produce efficient health care scheduling especially with the outbreak of new diseases, demographic changes in the population and strict budget from the government (Hans and Vanberkel (2012)). Research of health care problem focusing on the operating theatre (OT) scheduling problem has been widely conducted, for example by Boldy (1976), Smith-Daniels et al. (1988), Pierskalla and Brailer (1994), Yang et al. (2000), Litvak and Long (2000), Van Oostrum et al. (2008), Girotto et al. (2010), Fei et al. (2010b), and Ghazalbash et al. (2012).

Surgeries performed in the operating theatre will instantly affect patients' life and well-being which makes the OT a resource with the highest demand in a hospital. For a hospital, the OT account for more than 40% of its revenues and a similar large part of its costs HFMA (2005). Hence the significant requirement of equipment and labor for an OT, making it among the most expensive resources in a hospital. An efficient OT department thus significantly contributes to an efficient health care delivery system as a whole.

Determining minimum-cost staffing levels that provide adequate coverage to meet emergency demand is a strategic problem (Hans and Vanberkel (2012)). Due to costs, surgeries should not involve too much overtime as the cost of each additional hour in the OT is greater than the cost of a regular working hour.

In general, scheduling of patients consists of two elements. The first is that of elective patients where they are usually put into a waiting list for a period of time before they undergo treatment. The second element is that of non-elective patients or emergency patients that have to be treated immediately and pose a considerable challenge because these patients arrive at random.

Accordingly, suitable planning of operations matched with patients requirements of OT and surgeons will improve the efficiency of the OT. However, realistically, patients arrive one by one and the treatment required by each patient varies considerably. Moreover, scheduling decision must be made as patients arrive. Some patients requires the OT more urgently that the others and this should be reflected in the scheduling decision. We developed a way to systematically order the way in which the patients are treated.

With the above in mind, this study focuses on online schedule problem of emergency and regular patients, and reschedule for patients delayed at OT. We will develop heuristics and metaheuristics algorithm to obtain solutions for this online schedule problem.

1.2 Objectives

The objectives of this research are:

- To study the main elements of OT scheduling, namely the scheduling of advance booking for elective patients and the order to treat these patients according to patient's type.
- To develop optimisation models for scheduling and rescheduling of emergency patients and regular patients on a daily basis for single operating theatre and for a single day for multiple operating theatres running in parallel. The problem consists of patients arriving online into the system and testing our data with different algorithms. The models developed will be able to schedule emergency patients immediately and reschedule other patients by taking into account the variations in operation duration, while minimising cost and delay.

- To design optimisation algorithms using Heuristic's Technique, Local Search, Simulated Annealing (traditional and updated) and Branch and Bound method to improve scheduling and rescheduling process. The algorithms developed will be efficient to compute and able to produce good quality schedules.
- To test the models of OT scheduling using different parameters and data types by generating multiple data sets and assessing the robustness of our models and algorithms.
- To analyse computational test results to evaluate if our models achieve our aims of minimising costs and reducing delay and rescheduling.

1.3 Overview of thesis

The remainder of this thesis is organised as follows. Chapter 2 provides an overview of literature in Combinatorial Optimization Problem, health care problem focusing on OT scheduling in hospitals. Moreover, the technicality of online scheduling and the different methodologies for solving the problems in our research are explored. In Chapter 3, we discuss about problem statements, introduce and examine the different parts of the model, and review all the constraints in our problems. In addition, our design of algorithm using heuristic, local search and simulated annealing method is presented in chapter 3, together with computational experiment and results. In Chapter 4, we consider scheduling of multiple operation theatres in a single day. Finally, Chapter 5 contains the concluding remarks and future works.

Chapter 2

Literature Review

2.1 Introduction

In this chapter, we will explore some of the basic ideas and review the literature in relation to complexity theory, health care scheduling and efficient algorithms. First, a discussion of Combinatorial Optimization Problems (COP) and the related theory such as NP-hard problems will be presented together with the application of COP in health care. Next, we will delve wider into the hospital environment and review past research that is linked to some of the health care practitioners that use scheduling in their working environments especially nurses and physicians. We will also explore in detail the different research in theatre scheduling problem. We will then consider the literatures in relation to the method of solving the problems where certain algorithms and methods will be reviewed. This literature review will guide us to fill the gap in current research problems as well as to approach certain well-known problems. Certain methods that have been discussed in other problems are suitable to be extended to solve the problem of online scheduling. Besides that, the aims and constraints of certain scheduling problems such as

minimising tardiness, preventing delay and reducing costs can be used in relation to our research in operation theatre scheduling and rescheduling.

2.2 Combinatorial Optimization Problem

Combinatorial Optimization Problem (COP) is a part of optimization problem to find the best possible solution for the value of objective function (the function is either minimised or maximised) while certain constraints associated with the function are satisfied. To put it in simple terms, how do we obtain the best solution under certain conditions.

An instance of a COP can formally be defined as a tuple $(\mathbb{F}, F, f, f')$ with the following meaning:

$\mathbb{F}$: the solution space (on which F and S are defined),

F : the feasibility predicate (or the set of solutions),

f : the objective function f: $\mathbb{F} \rightarrow \mathbb{R}$,

f' : the minimum.

The feasibility predicate F induces a set:

S : the set of feasible solutions: $S = X \in \mathbb{F} : X$ satisfies F.

The goal is to find a feasible solution where the desired minimum of f is attained.

It is worth mentioning here that COP can be divided into two main components. The first component is the search component (among the solution space $\mathbb{F}$, find a solution from the set S) and the second is the optimisation component (among all solution in S, find one with the best quality). However, trivial cases might occur from these two components. On one hand, the COP might become a

pure search problem where it might be difficult to find a solution at all or that all solutions are the same in terms of quality (f is a constant function or that f' can be chosen arbitrarily). The other is that the COP becomes a pure optimisation problem where all the solution space can be feasible solutions or that it might be difficult to find solutions of high quality ($\mathbb{F} = S$).

Under complexity theory, optimisation problems can be divided into two classes, P and NP. P problem is the set of problems that can be solved in polynomial time on a deterministic Turing machine. On the other hand, NP problem (the NP stands for *non-deterministic polynomial*) is the set of problems that can be solved in polynomial time on a non-deterministic Turing machine. An important theoretical question in computer science (and by extension, Operations Research) is whether P = NP. This problem is yet to be proven and it is widely believed that P $\neq$ NP (Neumann and Witt (2010)). It is one of the millennium problems of The Clay Mathematics Institute which offers $1 million for the solutions to these problems (Cook (2017)).

An important subset of the NP problem is the NP-complete problem where a problem ρ is said to be NP-complete if (i) ρ is in NP and (ii) every problem in NP is reducible to ρ in polynomial time. Any problem that satisfies the second condition is said to be NP-hard (even if the first condition is not met) (Leeuwen (1990)). Another way to look at the second condition is that a problem is said to be NP-hard if it is *at least* as hard as any other problem in NP. The NP-hardness of a problem suggest that it is impossible to find an optimal solution without the use of an essentially enumerative algorithm, for which computation times will increase exponentially with problem size (Aarts and Lenstra (1997)).

As an alternative, an NP-hard problem means that it cannot be solved in polynomial time and if a sub-problem is NP-hard, then the main problem is also

NP-hard. However, NP-hard problems do not necessarily means they are NP-problems. Many COP are NP-hard problems as presented by Michael and Johnson (1979) and Ausiello (1999). COP has been applied in a wide variety of important fields such as telecommunications, computer network, transportation, planning and scheduling. A good first introductory reading on COP can be found in Wolsey and Nemhauser (1999), Pardalos and Resende (2002) and Leung (2004).

In the instance where an interesting COP has been proved to be NP-hard, there are several methods to approach and solve it. A perfect solution to the problem is usually difficult to find but a satisfactory solution is always beneficial. One can argue on the tractability of the problem (different problems have different computation time constraints) but the solution should be produced in reasonable computational time. If given enough computational resources, we can check for solutions in a longer time frame and it should produce at least one solution in a sensible amount of time. However, it would be better to use these resources for many problems rather than just one. There are many other methods to solve the problems, the two most widely used and discussed are heuristics and approximation algorithms.

Heuristics are algorithms that produce good quality solutions (not optimal or perfect), but sufficient enough for the intended purposes. Although heuristics might not explore the entire search space but it is able to produce good solutions within reasonable time. The combination of quality solution with reasonable time makes heuristics one of the preferred method in optimisation (Martí and Reinelt (2011)). There are many heuristic methods that can be used to solve any particular problem which are very different from each other in terms of the algorithmic approach or optimisation aim. Martí and Reinelt (2011) listed several classes of heuristics such as (1) Decomposition Methods, (2) Inductive Methods, (3) Reduction Methods, (4) Constructive Methods and (5) Local Search Methods. The constructive and local search methods form the foundations of the metaheuristic

procedures ((Aarts and Lenstra, 1997)). Constructive heuristics is the method of generating a solution to a problem starting from zero (or empty solution) and then extending the current solution until a complete solution is constructed. Local search heuristics (or local improvement) starts with complete or feasible solution of the problem and tries to improve it by local move. Each step of move will continue moving from one solution to another with a better value until there is no other way to improve it.

On the other hand, approximation algorithms are similar to heuristics but they require polynomial run times and do not ensure finding an optimum solution. Under this method, we accept that finding the optimum solution to the problem will be inaccessible, and tries to look for provable close to optimum solutions. Provable solution quality and provable run-time bounds are the reasons that make approximation algorithms mathematically robust when compared to heuristics (Williamson and Shmoys (2011)). An algorithm is a factor α approximation (α -approximation algorithm) for a problem if and only if for every instance of the problem it can find a solution within a factor α of the optimum solution. If the problem at hand is a minimization then $\alpha>1$ and this definition implies that the solution found by the algorithm is at most α times the optimum solution. If the problem is a maximization, $\alpha<1$ and this definition guarantees that the approximate solution is at least α times the optimum. Thus, a $\frac{1}{2}$-approximation algorithm for a maximization problem is a polynomial-time algorithm that always returns a solution whose value is at least half the optimal value (Williamson and Shmoys (2011)).

Historically, COP can trace its roots in economic problems, where the main aim is to plan and manage limited resources efficiently (Morton and Pentico (1993)). Since then, more technical applications that explain the economic needs of efficient resources planning were studied and modelled as combinatorial problem (Yu (2013)). Some examples in relation to economic planning is the sequencing

of machines, scheduling of production design and layout of production facilities. Nowadays, discrete optimisation problems are thriving everywhere. They are used in portfolio selection, investment planning, vehicle routing planning, scheduling of trains and air planes, assignment of workers and revenue management (Yu (2013), Paschos (2013)). The list is almost limitless. For the purpose of our research, we will focus on the problems in health care.

In recent years, where resource management are more inter-connected, the important problems in health care are also related to transportation (Schmid and Doerner (2010)), building management (Bowers and Mould (2002)), personnel management (Moz and Pato (2003), Gendreau et al. (2007)), medical equipments (Guinet and Chaabane (2003)), facilities and related services. In addition, another area that is studied in relation to health care is the delay problem in providing health care especially in surgeries. Delay in surgeries not only increases costs incurred to the hospital but also causes dissatisfaction to the patients. Motivated by the need to efficiently schedule elective surgeries while ensuring that the OT is available should it be needed for emergency, this research looks into the possible method of scheduling and tries to ascertain the best one to be implemented.

2.3 Scheduling

Given finite resources, the main aim is to ensure that all the resources are used efficiently and no wastage occurs. This is where scheduling and sequencing play important roles especially in production planning. Sequencing and scheduling are concerned with the optimal allocation of scarce resources to activities over time (Lawler et al. (1993)).

It will never be possible to talk about scheduling without mentioning the idea of sequencing since both are intertwined. In a more technical setting scheduling

and sequencing involve jobs to be completed by machine in a certain order. In technical term, scheduling is defined as assigning each operation of each job a start time and a completion time on a time scale of machine within the precedence relations. On the other hand, sequencing means that for each machine in the shop, one has to establish the order in which the jobs waiting in the queue in front of that particular machine have to be processed (Alharkan (2005)). In relation to project management, Morton and Pentico (1993) defined scheduling as:

> *Scheduling is the process of organizing, choosing, and timing resource usage to carry out all the activities necessary to produce the desired outputs at the desired times, while satisfying a large number of time and relationship constraints among the activities and the resources.*
>
> (Morton and Pentico (1993, p. 5))

As stated, scheduling will have the element of time involved in the decision-making process. The aim of all schedules is to perform all the necessary tasks with minimal time while using the least amount of resources. Some scheduling process looks into reducing the tardiness of the tasks while minimising costs.

2.3.1 Classical Scheduling

The basic assumption in classical scheduling theory is that processing times of jobs are constant. Herrmann (2006) stated that classical scheduling theory assumes a static, finite set of jobs waiting to be scheduled onto a production system and little consideration is given as to how this set may have arisen, its size, composition or whether it is static. In terms of the process, classical models have considered many different machine configurations from single stage, single machine to complex job shop configurations with multiple potential routes, and parallel non-identical machines at some or all processing stages.

The theory of classical scheduling can be summarised as follows (Herrmann, 2006, p.304) :

> *The classic sequencing/scheduling problem involves a set of jobs and a set of resources, where resources perform operations and each job requires one or more operations for successful completion. Job sequences for each resource (or equivalently, resource routes for each job) must be determined such that some combination of objectives is optimized and relevant constraints are satisfied. Common objectives include (i) makespan minimization, (ii) flow time minimization, and (iii) minimization of the number of tardy jobs. Example constraints include (i) job preemption, (ii) precedence relationships, and (iii) each resource can process at most one job at a time.*

Early on, researchers tried to find the best possible rule to schedule n jobs on a single machine when job j becomes available for processing at its release date r_j with processing time p_j. Usually, these jobs will have some weight attached to it. In essence, a job j usually has the following information associated with it (Leung (2004)):

Processing Time (p_{ij}) - If job j requires processing on machine i, then p_{ij} represents the processing time of job j on machine i. The subscript i is omitted if job j is only to be processed on one machine (any machine).

Release Date (r_j) - The release date r_j of job j is the time the job arrives at the system, which is the earliest time at which job j can start its processing.

Due Date (due_j) - The due date due_j of job j represents the date the job is expected to complete. Completion of a job after its due date is allowed, but it will incur a cost.

Deadline ($\overline{due_j}$) - The deadline $\overline{due_j}$ of job j represents the hard deadline that the job must respect; i.e., job j must be completed by $\overline{due_j}$.

Weight (w_j) - The weight w_j of job j reflects the importance of the job.

In addition, each job j will have its completion time, CT_j. Extending from that, the lateness of job j is defined as $Late_j = CT_j - due_j$ and the tardiness of job j is defined as $Tardi_j = max(Late_j, 0)$. The unit penalty of job j is defined as $U_j = 1$ if $CT_j > due_j$; otherwise, $U_j = 0$.

In terms of the objective function, the function that is optimised is always a function of the completion times of the jobs. The classical objective functions that are usually utilised are as follows (Leung (2004)):

Makespan (CT_{max}) - The makespan is defined as $max(CT_1, ..., CT_n)$.

Maximum Lateness ($Late_{max}$) - The maximum lateness is defined as $max(Late_1, ..., Late_n)$.

Total Weighted Completion Time ($\sum w_j CT_j$) - The total (unweighted) completion time is denoted by $\sum CT_j$.

Total Weighted Tardiness ($\sum w_j Tardi_j$) - The total (unweighted) tardiness is denoted by $\sum Tardi_j$.

Weighted Number of Tardy Jobs ($\sum w_j U_j$) - The total (unweighted) number of tardy jobs is denoted by $\sum U_j$.

One of the popular methods to classify scheduling problems was introduced by Graham et al. (1979) by using the $\alpha|\beta|\gamma$ notation. The α field describes the machine environment and contains a single entry. The β field provides details of job characteristics and scheduling constraints. It may contain multiple entries or

no entry at all. The γ field contains the objective function to optimize and it usually contains a single entry.

For single machine scheduling, a number of efficient algorithms have been developed to provide optimal solutions. The most notable are the work by Jackson (1955, 1956) and Smith (1956). Jackson (1955) proposed that a well-known problem of minimizing the maximum lateness on a single machine can be solved in $O(nlog(n))$ time by sorting the jobs in non-decreasing order of their due dates $due_1 \leq due_2 \leq ... \leq due_n$. This method of sequencing is called the earliest due date (EDD) rule or Jackson's rule. Note that Jackson's rule also produce the optimal schedule for the problem of minimising the maximum tardiness $Tardi_{max} = max\{0, Late_{max}\}$ on a single machine.

As stated earlier, some jobs have certain weight associated with it. With weighted jobs, the objective in the problems is to find a schedule to minimize $\sum_j w_j CT_j$ (average weighted completion time). The basic idea in this context is to minimise the completion time $\sum_j CT_j$ on a single machine with job j having a processing time p_j and that all jobs are available at time *0*. Ordering the jobs using the Shortest-Processing-Time (SPT) rule, where whenever a machine is free for assignment, assigning job with the smallest processing time among all unassigned jobs, will give an optimal schedule (Leung (2004)). With weighted jobs, the same idea follows where the objective is to minimise $\sum_j w_j CT_j$ where jobs are scheduled in non-decreasing order of the ratio p_j/w_j. This procedure also has a simple optimality rule as stated by Smith (1956) (known as Smith's rule).

As production processes evolve and become more complex over time, scheduling techniques also evolve to accommodate the true nature of the processes. This in turn creates other branches of scheduling theory and abandons the simplistic nature of the classical scheduling theory as well as considers more demanding

constraints that need to be achieved. However, the objectives of classical scheduling such as minimising makespan, minimising total weighted completion time and reducing tardiness remain the basis for many of the research.

Among the theories considered is how to schedule parallel identical machines and parallel unrelated machines. For parallel identical machines, there are m identical machines in parallel. Each job j requires a single operation and may be processed on any one of the m machines. Besides that, there are the concept of unrelated machines, where there are m machines in parallel, but each machine can process the jobs at a different speed. Machine i can process job j at speed s_{ij} . The time p_{ij} that job j spends on machine i is equal to p_j/s_{ij} , assuming that job j is completely processed on machine i .

In general, there are two decisions to be made in parallel-machine scheduling problems. First is to assign jobs to the machine and second is to determine the sequence of the jobs on each machine. Obtaining optimal solution to the problem of scheduling parallel machine is not easy and researchers usually employ heuristic algorithms to tackle this problem. In most heuristic algorithms, the list scheduling method is the method of choice (Shim and Kim (2007)). Under the list scheduling method, when a machine becomes available for processing a job, the jobs that can be processed on the machine at the time is selected based on a certain priority rule and scheduled on the machine. Similarly, when a job becomes available for processing, a machine is selected according to a priority rule among those that can process the job, and therefore the job is scheduled on the machine. Another thing to consider in scheduling is the idea of unrelated pair of assignments - if the completion time of jobs i and k remain unchanged when we reverse the order at which they are scheduled, then the jobs are unrelated.

List scheduling is the most popular scheduling approach since it is simple

and that any optimal schedule can be constructed by list scheduling with an appropriately chosen list. Besides, since list scheduling requires no knowledge of unscheduled jobs as well as of all jobs currently being processed, it is very powerful in online scheduling and especially so in online nonclairvoyance scheduling, in which it remains a dominant heuristic (Leung (2004)).

2.3.2 Online Scheduling

In off-line scheduling, the problems are all deterministic where all information regarding the jobs is available in advance. In this setting, the arrival of jobs and the processing time are known and fixed. However, in real life, scheduling decisions have to be made with incomplete or partial information.

In online scheduling, it is assumed that the arrival times of jobs are not known in advance (jobs may arrive at any time) but once a job arrives all of the data (i.e. processing times) are known. Usually, the main aim of the schedule is to minimise total completion time or to minimise total tardiness or idle time, but the aims varies from problem to problem. Online scheduling is very useful in our research since information is reviewed at all time. There are many examples of online scheduling in the literature such as Lu et al. (2003), Pruhs et al. (2004), Anderson and Potts (2004), Liu et al. (2011), Potts and Strusevich (2009), Tao et al. (2010) and Liu et al. (2011).

To put it into our scenario, we do not know which patients will arrive into the system. Some patients are referred by doctors, and some patients arrive from referral by the emergency department. Once a patient arrives, the time required in the OT is known since most hospitals currently use software designed by commercial surgical scheduling systems from electronic medical record vendor, where the prediction is based upon a moving average of previous cases, based on surgeon and procedure codes (Hosseini et al. (2015)). In some cases, the surgery is longer

than expected, which causes the last patient to be delayed to the next day. Some emergency patients need to be operated as soon as possible. At the end of each day, we have new information about the patient's priority (patient's priority is defined in the next chapter). Hence, our online scheduling planning procedure is to update the existing schedule of the OT daily based on the patient's priority.

2.3.3 Stochastic Scheduling

It is undoubtedly important that everybody involves in surgery should understand that although we can largely predict what will happen during surgery, there will always be some random elements present. Although surgeries may run smoothly, undoubtedly there are times where incidents might occurs, especially if there is any finding that might require some changes to the procedure that require extra time.

Whenever there are one or more random features in a model, the first thing to consider is whether it can be classified as a stochastic process. In particular, stochastic scheduling is a problem where scarce resources must be allocated over time to jobs with random features. In stochastic scheduling, the population of jobs is assumed to be known, whereas the processing times of jobs are random variables. Assignment of jobs and their processing times are modelled by specifying their probability distributions (which are assumed to be known). The actual processing times are known only after completion of the jobs.

In general, stochastic scheduling models can be classified into three broad categories: (i) models for scheduling a batch of stochastic jobs, (ii) multi-armed bandit models, and (iii) models for scheduling queuing systems (Niño-Mora (2009)). Our research is related to the third category where models in this category involves the design of optimal service disciplines in queuing systems, where the set of jobs to be completed, instead of being given at the start, arrives over time at random.

In on-the-day scheduling, strictly speaking, not all jobs (patients) arrive over time at random because the original schedule already existed. However, some patients (emergency patients) do arrive at random. Also, sometimes there is a delay that causes the scheduled patients to 'arrive' again into the system. The arrival of the emergency patients and the occurrence of delay (the time taken to complete the surgeries) are what we consider as random.The assumptions will be expounded in the next chapter.

Mancilla (2011) considered two problems of stochastic scheduling in OT. The first problem is in sequencing and scheduling of surgeries in a single operating room with the goal of minimising patient waiting time, OT idle time, and staff overtime. The second problem is sequencing a single surgeon in parallel OTs. While the surgeon is operating in one room, cleaning and set up procedures are being done in the other. The goal is to produce a sequence and schedule that minimise the surgeon's idle time, OT staff idle time, and staff overtime in each OT. In both problems, the approach taken is based on stochastic integer programming and sample average approximation. In our study, we will consider the problem of scheduling and re-scheduling patients in both single OT and multiple OTs with the aim of minimising cost and delay.

2.4 Scheduling in Hospitals

Hospitals are an integral part of human health with the advances in medical sciences. Hospitals are becoming more significant in providing services to patients in part because of the specialisation of the different branches of hospital care. Proper scheduling of all the components is vital in achieving their goals. Some of the popular components in scheduling problem are the scheduling of nurses, physicians, outpatient clinics and OTs.

2.4.1 Nurses

Currently, scheduling nurses is also related to OT and research into this area has been published in several papers on rescheduling approach in nursing problem (see Cheang et al. (2003) for a survey in this subject). In this area of research, the term roster is used to define the set of all nurse schedules of the unit. For example, Moz and Pato (2003) developed an integer multicommodity flow model for the problem of Rerostering Nurse Schedules. They test the performance using heuristic and integer optimizer package. The goal is to help head nurses in their rescheduling task. They found that it is possible to obtain an optimal schedule in reasonable computational time and that the heuristic results are also helpful when the optimisation procedure is too time-consuming.

Focusing on the same problem, Moz and Pato (2004) extended the research with harder constraints (such as labour contract rules and institutional requirement). Two new integer multicommodity flow formulations were developed where the first aims at optimising a flow in an n-cardinality level network while the second is the aggregation of the first. They proposed that the second formulation is more suitable for the general cases and have better computational results. Moz and Pato (2007) described constructive heuristics and applied several versions of genetic algorithms such as random keys encoding, crossover and mutation operators, and hybridisation to the nurse rerostering problem. Their main aim is to tackle the problem of rebuilding nurse schedules when unexpected staff absences arise. They performed tests with real data and concluded that good quality solutions can be achieved within the bound of the hospital.

From the nurses' points of view, their preferences in taking days off should also be considered when rescheduling decision is made. Bard and Purnomo (2005) solved the nurse preferences using a robust column generation procedure that combines integer programming and heuristics. Their result made an improvement

to the shift structure. They stated that good solutions can be obtained within a few minutes in the majority of cases.

Yeh and Lin (2007) proposed a simulation and genetic algorithm to adjust the nurses' schedule at a hospital emergency department without hiring additional staff to minimise the patients' queue time. They run a computational analysis and make comparisons and found that appropriate adjustment to the schedule will reduce patients' queue time which increases the quality of patient-care and satisfaction. This showed that a minor adjustment to the problem can greatly increase the quality of the schedule with existing resources without the need of additional staff which suggest that even with limited resources, a better schedule can always be achieved.

The main take from the problem of nurse scheduling is that when trying to create a roster, the main constraints mostly lie with the rule of labour contract, hospital administration's rule and the preference of the nurse. Heuristics and meta-heuristics have been researched and applied to this problem and have produced mixed success. Other methods such as genetic algorithm and branch and bound have also been successfully applied (Cheang et al. (2003)). We can deduce that method like genetic algorithm is suitable to the nurse scheduling problem since nurses have preferences that needs to be consider while producing the schedule. In our research, we do no take into account these factors and only consider the cost and type of patients. Our research will focus primarily on the patients.

2.4.2 Physicians

Unlike nurse rostering which has been extensively studied in the literature, maximizing satisfaction matters primarily in physician scheduling, as physician retention is the most critical issue faced by hospital administrations (Carter and Lapierre (2001)). In addition, while nurse schedules must adhere to collective

union agreements, physician schedules are more flexible and driven by personal preferences (Gunawan and Lau (2013)). There are several software packages that are available to address this problem but the benefits from these packages are still not fully reviewed.

Carter and Lapierre (2001) studied the problem of scheduling emergency room physicians. They interviewed physicians in order to understand emergency room scheduling problem. They stated that the real scheduling problem is difficult to assess because physicians' working conditions are usually based on informal mutual cooperation which is not documented. They proposed how to modify an existing scheduling rules to develop techniques which produce better schedule and reduce the time to create one.

Examining the same problem, Gendreau et al. (2007) proposed generic forms of the constraints encountered and reviewed several possible solution techniques that can be applied to the physician scheduling problems. The solutions considered are tabu search, column generation, mathematical programming and constraint programming. They discussed the suitability of each method depending on the specific conditions and discussed the problems in performing computational computations of solution techniques.

Gunawan and Lau (2013) analysed the problem of assigning master physician scheduling problem which included physician's full range of duties such as surgery, clinics and administration where the aim is to satisfy as many physicians' preferences and duty requirements as possible. It involved assigning physician activities to the appropriate time slots over a time horizon by taking into account rostering and resource constraints together with physician preference. They proposed mathematical programming models that represent different variants of this problem. The models are tested using real cases and randomly generated problem. However, if the problem cannot be solved optimally within reasonable time

by the exact method, heuristic algorithm (local search) is proposed which provided computational results of local optimum.

Unlike nurse scheduling, the physician scheduling problem requires more creative solution that resulted in many heuristics approaches. This is mainly because physician schedules are more flexible and primarily based on individual preferences since physician retention is the most crucial issue faced by hospital administration (Carter and Lapierre (2001)). This issue of preference will not be considered in our research since we will mostly only consider cost and patients' type in our scheduling constraints.

2.4.3 Outpatients

Operational Research has been exploited in health care settings since at least as early as 1958, when it was applied to outpatient departments where waiting times had been a frequent cause of complaints (Powell (2006)).

Some of the critical and important operational problems are in the hospital system. This system is related with patients' expectations of the services they received. Lorber (1975) suggested that medical personnel expect patients to be cooperative and undemanding while they adhere to the hospital routines. If patients do not conform to their expectation, they are labelled as 'problem patients' and may not be served properly. Although adequate care is given to these 'problem patients', the quality of service is minimum when compared to the superior care given to the patients that followed hospital procedures without any complaints. This situation should not occur and including these problematic patients into the modelling as a factor is unethical. Only the patients' type will be considered and not their behaviour towards medical personnel. In our research, we assume all patients will be receiving equal quality of care and unpunctual or no-show patients do not occur.

Ho and Lau (1992) investigated the various rules for scheduling outpatients appointments and their ability to minimise a weighted sum of medical personnel's and patients' idle-time costs. They showed that idle times are affected by the probability of no-show, the coefficient of variation of service times, and the number of patients per clinical session. Theoretically, an appropriate scheduling rule can be identified only if one knows the values of these parameters and the ratio between the medical personnel's and patients' idle-time costs. The rules are evaluated using simulation and the results are presented in the form of efficient frontiers, together with a simple procedure for identifying the best scheduling rule for given environmental-parameter values. This suggests that if we know or can approximate some of the factors such as service time or the number of patients per session, we can schedule more efficiently. Hence, we will study the service time (both expected and actual) and the average number of patients per day in our research and try to schedule both in one day and throughout the whole system efficiently.

Klassen and Rohleder (1996) addressed the problem of scheduling patients who call without knowing which type of patients may call later. The aim is to compare various scheduling rules in order to minimise the waiting time of patients and the idle time of the service provider. In order to reproduce the environment experienced by a typical family in an outpatient clinic, they interviewed the receptionists at two clinics which served as basis for the parameter that were incorporated in the model. The purpose of the interviews was not to obtain specific data but rather to understand what information is available to the receptionists and try to ascertain which factors influence performance of the system. The interview verified that the receptionists do have knowledge about the patients service time characteristics which is used to differentiate between patients and develop various scheduling rules. Based on that and prior research, they developed a simulation model of dynamic medical outpatient environment. They suggested that the best decision depends on the goals of particular clinic and the environment it

encounters. They also stated that scheduling clients with low standard deviation at the beginning of the session is one of the best rules. This finding suggests that by identifying the factors that might influence scheduling decision, a more realistic approach might be considered such as setting certain rules for the earlier patients might be beneficial in some situation while other rules in another situation.

Cayirli and Veral (2003) reviewed the previous literature on appointment scheduling for outpatient services and analysed appointment system which satisfied the objective of the system. They found that the practical effect of the appointment system is very minimal and they urged that the gap between theory and practice should be narrower in the future. For example, they suggested that future research should develop easy-to-use heuristics that can be utilised to choose the best appointment system for individual clinics and more empirical data should be used to identify probability distribution that represent actual service time. They also stated that there is a void in capturing the arrival patterns that incorporate unpunctual patients, walk-ins, and emergencies. There is also a need to study walk-in seasonality. In addition, they stress the need to use multiple measures of performance to evaluate the appointment system which include "fairness" as a factor and not only consider only cost as factor. These findings inform us that we need to develop an easy-to-use heuristic which incorporate the probability distribution of the arrival time and the service time. Most importantly, we will include the arrival of emergencies in our model.

Huang et al. (2012) described a design of appointment system in outpatient facilities where patient waiting time and waiting for physician idle time are considered and that it meets the scheduling policies without overbooking or double-booking. They proposed an approach bases simulation which contains several steps in obtaining the solution. First, data on the treatment times was collected to estimate time parameters and distribution for each visit type. Then simulations were run under any declared policy or constraints. Lastly the optimal schedule for each

visit type was developed and then the patient arrival schedule was constructed. They stated that the results can effectively reduce patient waiting time as much as 56% without significantly increasing physician idle time per patient and still allow physician to see and schedule the same number of patients per clinic session. However, one limitation for this approach is that patient no-show is not included in the model.

In light of all these research, our research will focus on creating a system which include patients' type as a scheduling factor and the need to amend the schedule to cater for the arrival of emergency patients that need immediate medical attention. We will also measure the performance of our method based on several criteria so that our method can address existing issues related to outpatients.

2.5 Theatre Scheduling Problem

In the previous section, we discussed some of the scheduling problems related to the hospital environment which focussed on the hospital staff. Now we will discuss the physical side of the hospital environment and go directly to the center of our problem which is scheduling OT. Scheduling OT for elective surgery is a complex task because many factors need to be taken into consideration such as surgeon priorities, availability of nurses and anaesthetists and the availability of suitable equipments. Surgeries are critical component in a hospital, not only for their costs but also because of their direct impact on patient's health. OT management focuses on maximizing the number of surgical cases that can be done on a given day and minimizing the required resources and related costs.

One of the major priorities of health care institutions are effective scheduling of OT while reducing their costs and maintaining high-quality services. Many

researchers such as Cardoen et al. (2010), Augusto et al. (2010), Guinet and Chaabane (2003), Jebali et al. (2006) and Dexter (2000) have carried out research in scheduling OT to improve surgical scheduling whilst achieving the objectives of the hospitals.

Early in the history of OT scheduling, most of the schedules are done manually. This task is performed by a single clerk in small hospitals and by an interdisciplinary team for large institutions. Later, in the late 1970s, optimal OT scheduling was developed in order to eliminate the need to cancel surgery because of insufficient surgical beds. Ernst et al. (1977) were among the first to use computer to produce OT schedule automatically. They developed a software program that includes a particular day for sorting patient's priority case, time and surgeon priority and room preference before assigning procedure's operation room. With this method, they managed to consistently reduce discord among OT personnel. Similar to this case, we will also consider patients' priority case, having proven that patients' case has been included in the early literature which shows that it is an important factor to consider.

The methods of advance scheduling patients and allocating scheduling on surgical demand was reviewed by Magerlein and Martin (1978). They define advance scheduling as the process for determining which patients are to be scheduled into a surgical suite on a particular day; allocation scheduling is the process of determining the sequence of cases within an operating theatre on a particular date, given that a slate of patients has been identified. Augusto et al. (2010) discussed in their literature review (citing Jebali et al. (2004)) that depending on the OT type, there are two surgery scheduling strategies, block scheduling and open scheduling. The policy of block scheduling is to establish a timetable called the Master Schedule and allocate the time slots to surgeons, group of surgeons or medical specialities. In the open scheduling policy, patients are scheduled without any speciality-related restriction. The authors considered the situation where patient

recovery is allowed in operating room due to a crowded recovery room in their open scheduling problem and developed a two-staged heuristic method to construct weekly surgery schedule considering the availability of surgeons and places in recovery room. In our research, we will consider the open scheduling policy in the sense that patients are scheduled without any restriction on the type of the surgery. This is because under the open scheduling system, the schedule is created prior to the day of surgery. The schedule specifies which surgeries are assigned to which OTs and their start time as opposed to the block-scheduling system, where either individual or groups of surgeons are assigned blocks of OT time in a periodic schedule (weekly or monthly). The surgeons may book cases into their assigned blocks subject to the condition that the cases fit within the block time (Gupta (2007)).

Dexter and Traub (2002) considered elective case scheduling where the aim is to maximise the efficient use of OT time by considering scheduling a new case into the OT using two patient-scheduling rules namely Earliest Start Time and Latest Start Time. Historical duration data is analysed to study the performance of the rules. The difference between the two rules were only a few minutes per OT and that depending on the objective, either one should be used with some restrictions in place. We will consider the feasibility of using any rule on the start time or decide to stick with one particular rule.

Gerchak et al. (1996) discussed the problem of planning for elective surgery when the capacity of OT is shared with emergency surgeries. They discussed the problem of admitting elective surgeries when the OT is mostly used by the emergency department. Admitting too many elective surgeries might worsen the patient's health, exceeding the hospital capacity and producing less productive work. They did not consider patient's priority in their model and their work is extended by Min and Yih (2010b) which considered the effect of patient priority on the surgery schedule.

Min and Yih (2010b) stated that patient priority should be considered in the surgery schedule and that insufficient consideration may result in an ineffective schedule. They also suggested that the higher number of emergency patients arriving and the duration of surgery also contribute to the inefficiency of surgery schedule. In their paper, a set of patients are selected from a waiting list at the beginning of each period and the schedule is produced based on priority. They made the decision of the number of patients to be scheduled based on trade-off between the cost of surgery delay and the cost of overtime on that day.

Weinbroum et al. (2003) conducted an efficiency study of OT use in a metropolitan public hospital. They stated that the time OT not used for given patients amounted to 5 days in a month which can be reduced. They attributed several reasons as the cause of OT being unused such as surgeon unavailability and inappropriate patient preparation. However, the main cause for time wastage is the unavailability of room or staff (59% of the time). Administrator will cancel the use of OT because of insufficient nursing or medical staff or because the OT was occupied by emergency operations. They also stated that inaccurate surgical time prediction also contributed to inefficiency. This suggests that we need to consider how emergency arrival will disrupt a current schedule and the disruption caused by inaccurate duration planning.

Bowers and Mould (2004) suggested that large uncertainty in orthopaedic care means that much of the theatre time is not utilised. They developed simulation to explore the balance between maximising the utilisation of theatre sessions, avoiding too many overruns and ensuring a reasonable quality of care. They proposed that if patients are willing to accept the possibility of their treatment being cancelled (and also the probability of earlier treatment), greater throughputs can be achieved. They also performed several approximations as alternatives to the full simulation. They stated that although the results depends on assumptions about the patient selection criteria, the sensitivity is not great and the mixing

of elective and emergency patients appears to be robust. They suggested that the simpler model is better to be implemented and more appropriate for practical planning purposes. Similarly, we will focus on maximising the use of OT without too many overtime and reducing costs of delays.

Calichman (2005) suggested that by analysing hospital's bed use data, cancellations of surgeries can be minimised with a schedule that uses all available surgical beds. The schedule is obtained by arranging surgical procedures on different days to minimise and balance the number of beds required each day. The schedule obtained gained full use of the hospital's bed and managed to prevent up to 18 cancellations each week and increased revenue by 3%. They suggested that the key for best possible OT schedule is to use the historical relationship that exists between each surgical category and its length-of-stay distribution that is hidden in the hospital data. This idea of using historical data will certainly be a good way to assess the outcome of any method and we will be using generated data based on real life situations to test our model.

Bhattacharyya et al. (2006) also considered the option of having specific OTs for emergency patients only to reduce night-time cases and improve OT flow but they only focused on the orthopaedic unit of a hospital where no elective cases were scheduled in the unbooked trauma OT. They collected OT data time on dynamic hip screw and closed femoral nailing (two common surgical cases) and reviewed data on waitlist cases, surgical time, anaesthetic time, OT utilisation and surgical compliance by retrospective analysis for two 1 year periods before and after the unbooked trauma OT was introduced. They suggested that with the availability of the unbooked trauma OT, the operating suite flow improved by significantly reducing the number of hip fracture cases performed at night, significantly reducing the number of orthopaedic waitlist cases that began after 5 PM, significantly reducing the number of elective cases which were bumped by emergencies, and decreasing over-utilization. They also proposed that hospitals

should establish or expand orthopaedic trauma block time. This idea of having specific OT for use in case of emergency or unseen problems is good because when emergency happens, any surgery can be performed immediately without the need to disrupt existing schedule.

Wullink et al. (2007) analysed the best way to reserve OT time for emergency surgery by comparing two approaches: assigning specific OT for emergency patients and evenly reserving capacity in all elective OTs. They modelled real situation using a discrete event simulation where the main outcome measured are waiting time, staff overtime and OT utilisation. They suggested that the second approach, with emergency capacity allocated to all elective OTs, surpasses the first approach on all outcome measures. They stated that the policy of allocating OT capacity for emergency surgery to elective OTs requires the OT department to be flexible and the OT to be equipped for all kinds of emergency surgery. This shows that unlike Bhattacharyya et al. (2006), with emergency capacity allocated to elective OT, better performance can be achieved and the only setback is the need for OT department to be flexible and all OTs be equipped with the necessary tools for all emergency surgeries.

Lamiri et al. (2008) proposed a stochastic OT planning model that specifically include both elective and emergency patients which minimises elective patient related costs and overtime costs of OT. They assumed the operating time of all elective cases are known and deterministic and considered the uncertainty in emergency demand in their Monte Carlo optimisation method which includes Monte Carlo simulation and Mixed Integer Programming. The computational results suggested that the optimisation method provides solutions that converge to a real optimal. They suggested that their planning model is useful for hospitals using a blocked advanced scheduling system. Comparatively, we will consider approaching the problem of emergency uncertainty by using probabilistic method to model the

arrival of emergency but we will improvise by considering the probabilistic nature of elective operating time in that actual duration might differ from planned duration.

Wachtel and Dexter (2009) analysed the data from two surgical suites to study the various factors that contributed to tardiness. The study found that tardiness is related to the total duration of preceding cases. Tardiness per case increases as the day progresses (because the total time increases) but for cases which were scheduled 6 hours after the day starts, tardiness declines. They suggested that tardiness is influenced by the total duration of preceding cases, expected under-utilised time or over-utilised time at the end of the day and case duration bias. The finding that total duration of preceding cases increases tardiness suggests that total duration for the day should not be too long and that the biasness in case duration can be prevented if the model included a realistic approach to time.

Min and Yih (2010a) considered patient priority in scheduling elective surgery with limited capacity. The trade-offs between the costs for overtime and costs of postponement is analysed using stochastic dynamic programming model in deciding the number of patients to include in the schedule. The results showed that improvement is achieved when patient priority is considered in the schedule. By considering patient priority, necessary surgery can be performed to those that really need it first. Our research will primarily focus on similar idea where we will include patient priority into our scheduling decision and compare different methods to see which one is the best to reduce costs and rescheduling.

Since most research is focused on improving the schedule of OT, Basson et al. (2006) conducted a study looking into surgical case cancellations. They conducted a retrospective review of OT records to identify the causes of cancellations. They

then conducted a stratified case control-control study of patients records to identify pre-existing factors that predict non-appearance of patients. A multivariate analysis suggested that non-appearance can be predicted from the patient non-compliance with clinic visits and other clinical procedures. They suggested that non-compliant patients should be booked at the end of the OT day, when cancellation effects on the OT flow is minimal. Our research does not include any case of patient's non-appearance since any patient that requires surgery will be at the hospital ready for the surgery and only rescheduling will occur.

Fei et al. (2010a) used open scheduling strategy to explore the possibility of improving the efficiency of OT. They developed a two-staged heuristic method to construct weekly surgery schedules with an open scheduling strategy by taking into account the availabilities of both surgeons and places in the recovery room. First, the planning problem is solved to give the date of surgery for each patient by a set-partitioning model solved by a column-generation-based-heuristic taking into account the availability of OT and surgeons. Secondly, a daily scheduling problem regarded as a two-staged hybrid low-shop model is devised to determine the sequence of operations in each OT in each day, taking into account the availability of recovery beds. The second stage is solved by a hybrid genetic algorithm, using a tabu search procedure as the local improvement operator. The results are compared with several actual surgery schedules and the proposed method has less idle time between surgical cases, higher utilisation rate and produces less overtime.

However, if there are unexpected numbers of emergency, rescheduling is necessary to update the schedule (Vieira et al. (2003)). Resheduling in general has been an area where many researches have been conducted especially in industries that are heavily related to scheduling such as airline industry (Clausen et al. (2010)) and railway industry (Narayanaswami and Rangaraj (2011)). In the manufacturing practice, Hall and Potts (2004) considered rescheduling problems where the aim is to minimise some cost objective when some new jobs need to be inserted

into the schedule without much disruption. They considered two classes of model. The first is to minimise the scheduling cost of all the jobs and the second is to minimise a total costs objective (both original cost measure and the cost of disruption). They provide either an efficient algorithm or a proof that such algorithm is unlikely to exist.

Erdem et al. (2012) developed a Mixed Integer Linear Programming (MILP) for rescheduling elective patients upon the the arrival of emergency patients. Two types of clinical units are considered which are the OTs and post-anaesthesia care units (PACUs). The model considers the overtime cost of the clinical units, the cost of postponing or preponing elective surgeries, and the cost of turning down emergency patients. In certain cases when it is hard to find an optimal solution, genetic algorithm is developed to efficiently obtain the approximately optimal solutions. They suggested that the two methodologies should be used jointly to provide good and timely decisions in admitting emergency patients and rescheduling elective patients.

All these literatures have provided many ideas for improvement for future research. The main gap that we see is the idea of rescheduling patients when emergency occurs during the day while minimising disruption on the elective schedule. Like several of the literature, several criteria should be included in the scheduling and rescheduling decision. Another central aspect is any method used should be computational efficient so that when emergency occurs, a new schedule can be made in a short amount of time.

2.6 Methods of Solving

In this section, we will discuss some of the methods that are related to our research. The principal theory of each method will be reviewed in general and where

appropriate how the theory is used to construct our solutions will be discussed. It should be stated clearly here the assumptions that underline our research. Our research is based on the OT in general where non-life-threatening elective surgeries are usually performed and not the OT in accident and emergency (A&E) where some cases might need attention immediately. Emergencies in our research are those that are non-fatal conditions and can be delayed.

2.6.1 Heuristics

The term heuristics stems from the Greek word *heuriskein* which means to find or discover. It is used in the field of optimization to characterise a certain kind of problem-solving methods (Martí and Reinelt (2011)). When we are unable to find a perfect solution, an approach that produces good quality solutions (but not optimal) is sometimes used. Such method is usually called a heuristic. Heuristics are especially suitable for problems arising in practice. The following is considered as a prime description of heuristics.

> *A heuristic technique (or simply heuristic) is a method which seeks good (i.e.near-optimal) solutions at a reasonable computation cost without being able to guarantee optimality,and possibly not feasibility. Unfortunately, it may not even be possible to state how close to optimality a particular heuristic solution is.*
>
> (Rayward-Smith et al. (1996, p.5))

Several heuristics methods have been proposed in the literature such as local search, hill climbing, neighbourhood search and gready algorithm; and the theory behind each method is explained in Burke and Kendall (2005).

Guinet and Chaabane (2003) considered a medium term horizon OT scheduling problem where each patient needs particular surgical procedure which defines

the human (surgeon) and material (equipment) resources to use as well as the intervention duration. They proposed a two-step solution to this problem. First, an OT planning is defined by assigning patients to OTs over the horizon. Next, each loaded OT is scheduled individually in order to synchronise the various human and material resources used. The problem is solved heuristically by proposing an assignment model with resource capacity and time-window additive constraints. The primal-dual heuristic integrates release and due date constraints with limited capacity constraints which optimised OT overload and patient waiting time.

Krempels and Panchenko (2006) considered the problem of semi-automated dialog-based system where a human planner is involved in the scheduling decision. They proposed a heuristic technique to create proposals for the schedule. The planner then acts as a sensor with the responsibility to identify changes as they occur and integrate his knowledge and decision-making competence in the planning proses. The planner will consider the interest and preference of the personnel involved in the schedule, which will reduce non-acceptance. Introducing a human planner in any model is aimed at making necessary changes that only a human can execute but in our research, no human planner will be considered.

Beliën (2007) built the models with stochastic numbers of patient for each operating room block and a stochastic length of stay for each operated patient and developed a number of mixed integer programming based heuristics and a metaheuristic (simulated annealing) to minimise the expected total bed shortage. Again the themes of reasonable computational time and stochastic nature of the factors will play an important part in our research.

Pradenas et al. (2012) considered the problem in weekly surgery scheduling and surgeon assignment in a public hospital. Their study was divided into two parts: first, surgeries were scheduled with support from a multi-knapsack mathematical model and then surgeons were assigned using a search method based on

chronological backtracking heuristic that possessed constraint programming property. They argued that the heuristics was selected because backtracking heuristic guarantees a solution or determines that there are no solutions. They showed that the heuristic obtained a feasible solution for every assignment and on certain cases, the algorithm had 38.90% more surgeries compared to manual procedure.

In this thesis, a 'good solution' is one that does not cause too many patients being delayed and minimises the costs of OT. The objective functions considered are discussed later in the thesis. In terms of computational time, we do not impose any limitation since our main objective is the minimisation of delays and costs. However, if the time taken is too long, then perhaps a more efficient method that shows significant improvements can be developed later.

2.6.2 Metaheuristics

Metaheuristic is a general class of algorithms and techniques that are able to provide a sufficiently good solution to an optimisation problem. The aim of using metaheuristic techniques is to explore the search space and find a near-optimal solution. There are many examples of metaheuristic such as local search, simulated annealing, tabu search and genetic algorithms. Generally, metaheuristics produce higher quality results than simple heuristics.

2.6.2.1 Local Search

Local Search is a method that explores the space of possible solutions sequentially. The algorithm performs a series of moves on the initial solution to find a local optimal solution. These moves are designed based on neighbourhood structure. In each iteration, if a better solution exists, then it is selected as a current

solution. This procedure is continued until no better solutions can be found in the neighbourhood of the current solution.

One of the most utilised local search method is the Variable Neighbourhood Search. Burke and Kendall (2005) suggest the following as the basic Variable Neighbourhood Search:

Initialisation

Select the set of neighbourhood structures N_l, for $l = 1, ..., l_{max}$, that will be used in the descent; find an initial solution x (or apply the rules to a given x);

Repeat the following sequence until no improvement is obtained:

(1) Set $l \rightarrow 1$;

(2) *Repeat* the following steps until $l = l_{max}$:

(a) *Exploration of neighbourhood.*

Find the best neighbour x' of x ($x' \in N_{l(x)}$);

(b) *Move* or not.

If the solution x' thus obtained is better than x, set $x \rightarrow x'$ and $l \rightarrow 1$; otherwise, set $l \rightarrow l + 1$;

In our research, based on the list of current patients, we swap a pair of patients based on certain conditions and then calculate the total cost of OT because of the swapping. If the cost is below the cost of the current list, we accept the swapping as a current solution. We iterate this procedure until we go through all the patients list. The conditions considered and the full method is discussed in chapter 4, section 3.6.

2.6.2.2 Simulated Annealing

The term simulated annealing (SA) originated from a process of cooling molten metal. The procedure has been developed initially as an algorithm to simulate the process of cooling and crystallization of materials in a heat bath, known as

the annealing process as discussed by Metropolis et al. (1953) in thermodynamics. Pirlot (1996) discussed that the idea of simulated annealing originated from thermodynamics and metallurgy, whereby molten iron is cooled slowly enough resulting to its tendency to solidify in a structure of a minimal energy. This annealing process is the same as our local search strategy of updating the current solution by a solution x randomly chosen in its neighbourhood if it is accepted. Then gradually, the temperature is decreased which means that one becomes more selective in accepting new solution.

Simulated annealing is a type of local search algorithm. Just like a local search (descent algorithm), it starts with an initial solution mostly chosen at random. A neighbour of this solution is then generated and the change in cost is calculated. If there is a reduction in cost, the current solution is replaced by its neighbour, otherwise the current solution stays. This process is repeated until no further improvement is found in the neighbourhood of the current solution and the algorithm terminates at a local minimum. Although a descent algorithm is simple and quick to execute, the disadvantage of the method is that the local minimum found may be far from any global minimum (Eglese (1990)). This can be countered by starting the algorithm at different initial solutions and choose the best of the local minimum found.

In simulated annealing, instead of sticking with this strategy, it avoids becoming trapped in a local optimum by sometimes accepting a neighbourhood move which increases the value of f. The acceptance or rejection of an uphill move is determined by a sequence of random numbers, but with a controlled probability. The probability of accepting a move which causes an increase Δ in f is called the acceptance function and is normally set to $e^{-\Delta/K}$, where Δ is the difference of total cost between a pair of patients we plan to swap (new costs - current costs), and K is temperature.

These choices must be made for any implementation of SA and constitute the annealing or cooling schedule, (i) the initial value of the temperature parameter K, (ii) a temperature function, $K(t)$, to determine how the temperature is to be changed, (iii) the number of iterations, N(t), to be performed at each temperature, and (iv) a stopping criterion to terminate the algorithm.

Eglese (1990) proposed the following algorithm for the general simulated annealing procedure.

Algorithm 2.1 Simulated Annealing Algorithm

1: Select an initial state $i \in S$;
2: Select an initial temperature $K > 0$;
3: Set temperature change counter $k = 0$;
Repeat
4: Set repetition counter $n = 0$;
Repeat
5: Generate state j, a neighbour of i;
6: Calculate $\Delta = f(j) - f(i)$;
if $\Delta < 0$ then $i := j$
else *if* $random(0, 1) < e^{-\Delta/K}$ *then* $i := j$;
$n := n + 1$;
until $n = N(k)$;
$k := k + 1$;
$K := K(k)$;
until stopping criteria true.

2.6.2.3 Other Methods

There are also other methods widely used in solving combinatorial problems. Here we discuss several of these methods.

1. Tabu Search

Tabu search is a method proposed by Fred Glover in 1986 in order to allow hill climbing to overcome local optima. The main idea of tabu search is to continue

the search whenever a local optimum is encountered by allowing non-improving moves. A tabu list records recent history of the search, which are referred to as tabu moves. Moving back to these previously visited solutions is forbidden under tabu search.

Hertz et al. (1995) summarised the tabu search algorithm as follows.

Algorithm 2.2 Tabu Search Algorithm

1: Choose an initial solution $i \in S$. Set $i* = i$ and $k = 0$.
2: Set $k = k + 1$ and generate a subset V^* should consist of solutions that either satisfy the tabu conditions or at least one of the aspiration criteria hold.
3: Choose a best j in V^* and set $i = j$.
4: If $f(i) < f(i*)$ then set $i* = i$.
5: Update tabu and aspiration conditions.
6: If a stopping condition is met then stop. Else go to Step 2.

2. Large Neighbourhood Search

Large Neighbourhood Search (LNS) is a method that uses heuristics to explore a complex neighbourhood. In LNS, an initial solution is gradually improved by repeteadly destroying and improving the solution. LNS belongs to the class of heuristics known as Very Large-Scale Neighbourhood Search (VLNS). Using large neighborhoods makes it possible to find better candidate solutions in each iteration and hence traverse a more promising search path (Pisinger and Ropke (2010)).

In general, the larger the neighbourhood, the better is the quality of locally optimal solutions, and the greater is the accuracy of the final solution obtained. At the same time, the larger the neighbourhood, the longer it takes to search the neighbourhood at each iteration (Ahuja et al. (2000)).

3. Ant Colony Optimisation

Ant Colony Optimisation (ACO) is a metaheuristic optimization method and a part of the Swarm Intelligence approach that search for optimal path in the graph based on behaviour of ants seeking a path between their colony and source of food. At the core of this behavior is the indirect communication between the ants by means of chemical pheromone trails, which enables them to find short paths between their nest and food sources (Blum (2005)).

The characteristic of ACO algorithms is their explicit use of elements of previous solutions. The essential trait of ACO algorithms is the combination of a priori information about the structure of a promising solution with a posterior information about the structure of previously obtained good solutions (Maniezzo et al. (2004)).

When searching for food, ants initially explore the area surrounding their nest in a random manner and leave a chemical pheromone trail on the ground. When choosing their way, they tend to choose paths marked by strong pheromone concentrations. When an ant finds a food source, it evaluates the quantity and the quality of the food and carries some of it back to the nest. During the return trip, the quantity of pheromone that an ant leaves on the ground may depend on the quantity and quality of the food. The pheromone trails will guide other ants to the food source (Dorigo and Blum (2005)).

In general, the ACO approach attempts to solve an optimization problem by iterating the following two steps (Blum (2005)):

- candidate solutions are constructed using a pheromone model, that is, a parameterized probability distribution over the solution space;
- the candidate solutions are used to modify the pheromone values in a way that is deemed to bias future sampling toward high-quality solutions.

4. Evolutionary Algorithms

Evolutionary Algorithms (EA) try to solve complex problems by following the processes of Darwinian evolution. The underlying idea in EA is that given a population of individuals, the environmental factors cause natural selection (survival of the fittest), which causes a rise in the fitness of the population.

The theory of evolutionary algorithm can be summarised as follows (Eiben and Smith, 2003, p.15);

> Given a quality function to be maximised, we can randomly create a set of candidate solutions, i.e., elements of the function's domain, and apply the quality function as an abstract fitness measure - the higher the better. Based on this fitness, some of the better candidates are chosen to seed the next generation by applying recombination and/or mutation to them. Recombination is an operator applied to two or more selected candidates (the so-called parents) and results one or more new candidates (the children). Mutation is applied to one candidate and results in one new candidate. Executing recombination and mutation leads to a set of new candidates (the offspring) that compete - based on their fitness (and possibly age) - with the old ones for a place in the next generation. This process can be iterated until a candidate with sufficient quality (a solution) is found or a previously set computational limit is reached.

Eiben and Smith (2003) stated that there are two fundamental forces that form the basis of evolutionary systems:

- Variation operators (recombination and mutation) create the necessary diversity and thereby facilitate novelty.
- Selection acts as a force pushing quality.

Algorithm 2.3 Evolutionary Algorithm

1: Generate an initial population $P(0)$ and set $i = 1$;
2: Evaluate the fitness of each individual in $P(0)$;
Repeat
3: Generate offspring from the parents using *variation operators* to form $P(i)$;
4: Evaluate the fitness of each individuals in $P(i)$;
5: Select parents from $P(i)$ and $P(i-1)$ based on their fitness;
6: $i = i + 1$;
until stopping criteria are satisfied.

2.6.3 Branch and Bound

Branch and bound is one of the methods of exact algorithm. It is a systematic method for solving discrete and combinatorial optimization problems. The method searches the complete space of solutions for a given problem looking for the best solution and is used to find a value x that maximizes or minimizes the value of an objective function $f(x)$. It is able to compute a lower and an upper bound on the optimal value over a given region. However, if the region is too big, exploring all the possible alternatives to find the best solution might not be computationally viable.

> *The rationale behind the branch and bound algorithm is to reduce the number of alternatives that need to be considered by repeatedly partitioning the problem into a set of smaller sub-problems and using local information in the form of bounds to eliminate those that can be shown to be sub-optimal.* (Burke and Kendall (2005, p.24))

The basic principle of a Branch and Bound algorithm with objective function $f(x)$ can be considered as follows:

1. The solution space can be regarded as a tree where the leaves are the set of all possible solutions.

2. Start from the root, then split the search space into smaller spaces and optimise $f(x)$ on these spaces. The method of splitting the spaces is called "*branching*". Branching alone can lead to the optimal solution but as stated earlier, it might not be computationally viable.

3. To improve the method, the algorithm calculates a "*bound*", i.e. the best solution from the sub-tree below the branch (or node). The bound is re-calculated at every decision point of the algorithm. The best solution at each decision point is considered as the benchmark.

4. The benchmark is then compared with the best solution at each level. If the solution is worse than the benchmark, the whole sub-tree can be discarded. this is known as "*pruning*" the search space, i.e. eliminating the candidate solutions that will not contain an optimal solution.

5. The *pruning* method usually reduces the search space by a large amount (depending on where it occurs - the closer to the root the bigger).

6. Keep searching the remaining search space until the optimal solution is found.

2.7 Discussion

Under the widely accepted assumption that if P $\neq$ NP, it is impossible to have algorithms that satisfy these three conditions of (i) finding optimal solution; (ii) in polynomial time; (iii) for any instance. One or more of these requirements must be relaxed in approaching an NP-hard problem. It is a tough problem that requires careful formulation in order to be solved. As we have seen, there are many

methods that can be used to solve these problems and each method has its own advantages and disadvantages.

As we have seen in this chapter, the problem of scheduling in health care management can be divided into several categories. In each category, there are many approaches to solve the problems encountered each with a specific set of aims and objectives. One common theme across the literature is that each problem is unique and requires a specific solution to be resolved. There is no one optimal or best method to tackle all the problems encountered. Health care providers need to consider each decision that are made to ensure that the quality of services provided are the best at the lowest possible costs while taking into account the needs and preferences of medical personnel involved.

Many researches have been conducted in the field of scheduling in hospitals especially in scheduling nurses, physicians and surgeons. Several researches have also been carried out in scheduling walk-in patients in clinic and outpatient department. Research predicting walk-ins, no-show, and service time has been invaluable but research that looks into future arrival patterns and the disruption caused by emergency arrival is also of value. Although online systems are more common in practice but studies that consider realistic arrival patterns and future arrivals are still needed to fully understand how the systems behaved. Our study in particular tries to study how the system behaves when the arrival of emergency patients disrupts current schedule.

In terms of performance measures, there are several criteria that are most commonly used such as patient waiting time, system overtime, number of patients seen and number of delayed patients which will also be used in our research. We will also add another important measure which is cost since a key trade off in accepting patients is the between service provision revenue and costs that are incurred from patient waiting time and provider overtime (Ahmadi-Javid et al.

(2017)). Ahmadi-Javid et al. (2017) also argues that factors that should be taken into account in scheduling patients are the demand of each patient group, priority level, no show probability, revenue from each patient group, and preferences of patients and physicians. For our research, we will consider priority level as one of the defining factor in scheduling decision.

With regard to modelling approaches and solution methods, we will utilise stochastic optimization in order to deal with the uncertainties inherent in patients scheduling especially in predicting the random arrival of emergency patient and the discrepancy between planning duration and actual duration of the surgeries. Due to the inherent complexity of the problem, we will mainly use heuristics and metaheuristics as our solution methods. The heuristic method is utilised because it is more flexible than the exact method thus allowing the incorporation of conditions that are difficult to model and if used as part of a global procedure guarantees to find the optimum solution of a problem Martí and Reinelt (2011)). In addition, metaheuristic is utilised because of its ability to guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search spaces using learning strategies to structure information in order to find near-optimal solutions (Kelly (1996)). The methods employed will be discussed in the relevant chapters. In order to ascertain the effectiveness of our methods, we will use random data simulation to generate relevant data and feed it to our model.

In conclusion, our research will develop an online scheduling model in the presence of the disruptive factor of emergency arrival while incorporating patient priority level, random service time, random arrival of emergency, and the cost associated with overtime usage and delays of patients. We approach this problem by employing heuristic, metaheuristic methods and specialist integer program software. We then proceed to test our model by simulating several random data and comparing the performance of our methods.

Chapter 3

Operating Theatre Scheduling Problem

3.1 Background

Scheduling of OT is related to the scheduling of patients since it is based on the expected duration of the surgeries performed. Duration of the surgeries will be stochastic in nature when patients arrive because when we schedule the patients into the different slots in the hospital system, the actual duration of the surgeries usually differ from the planned time.

Another aspect that needs to be considered is that new patients arrive into the system at all times, either regular patients or emergency patients. Regular patients have a large time window since we can schedule them without the urgency. On the other hand, emergency patients need to be scheduled as soon as possible because they need immediate attention and any delay will cause patient dissatisfaction. The scheduling of these emergency patients usually causes delay in the system.

There have been many research in relation to the delay caused by the arrival of emergency patients. This problem is usually caused by the sharing of resources between elective surgeries and emergency surgeries especially the OTs.

In our research, patient priority is one of the defining factor in producing the schedule. A list of schedule (for elective surgery) already exists and the problem here is to produce a new schedule when there is a delay and arrival of emergency patients.

3.2 Problem Definition

We consider two types of important patient priority arriving online. Emergency patients and regular patients arriving into the system after other patients have been scheduled. We allocate the patients depending on their urgency priority with operation time planned in advance but actual operation time are not known. The planning and scheduling always reveal the time (assumed) for each patient, but when the original schedule changes due to disruption, we need to reschedule the patients in the system.

We allocate all patients and reschedule delay patients depending on their cases using the heuristic method in time horizon every day. For an emergency case, the time scheduled will be as soon as possible in the time horizon (Figure 3.1). Patients go through the operation process on a fixed day and fixed hour depending on the planning duration.

Occasionally, actual duration for each patient is different compared to the planning duration and emergency patients do arrive into the system. Therefore, other patients must be reschedule at the end of the day. The decision whether to extend the usage of OT or reschedule the last patient depends on the costs (cost of overtime vs cost of reschedule). In order to avoid a lot of delay from occurring and

patients being delayed several times, high penalty costs are imposed for long delays. Hence, the model will try to schedule them as soon as possible to prevent high overall costs. We do not impose any constraint associated with either delay time or the number of delays. This means that patients can be delayed for several times as long as the model deemed the costs is acceptable especially when compared to the overtime cost. It will be highly costly if patients are delayed for too long and cause an inefficient scheduling of operation room. The goal are to ensure patients' safety and optimal patient outcome, to decrease patient delays, to maximise the efficiency and minimise the cost of OT.

For example, in Figure 3.1, we have patients $P_1, ..., P_5$, slotted for operation in Day 1 until the end of the time horizon for the day, T_d. At the end of the day, emergency patients and new patients arrive into the system. Before the next day, we must prepare a new schedule with a new list of patients. Rearranging the list will depend on the patient priority. In situation (i) of Figure 3.1, we set a duration of T_e, the slot of time for emergency patients or delayed emergency patients at the beginning of the next day. When this happens, the slots for regular patients will be delayed and the patients at the end of the time horizon will be delayed to different days. Another situation that can occur is that emergency patients do not arrive but the actual duration of the surgeries might be longer than the planned duration because of their stochastic nature. In situation (ii) of Figure 3.1, we can see that the actual duration for patients P_1, P_2 and P_3 are longer than the planned duration, hence the slots for patients P_4 and P_5 are no longer available hence they need to be rescheduled.

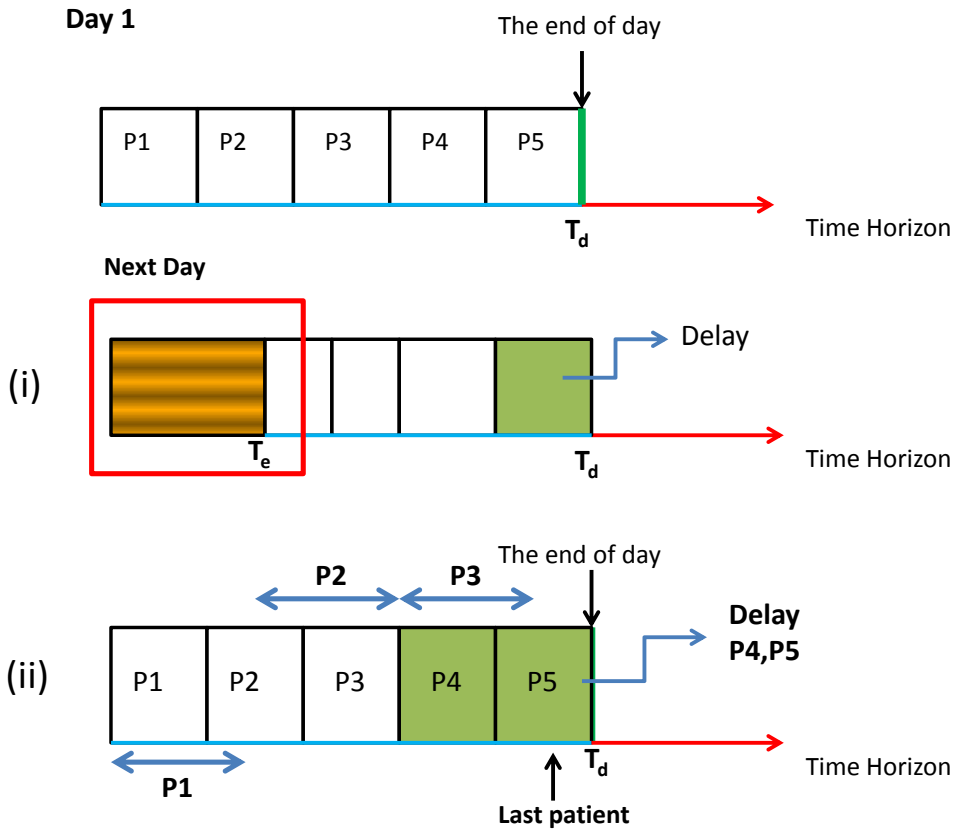

FIGURE 3.1: Delay situation

Besides rescheduling the patients to different days, we can also decide to perform the surgery with overtime usage of the OT. The decision is based on the comparison of the overtime cost and rescheduling cost. Before that, we need to make sure that the planning duration of the last patient will suit with the OT overtime horizon. If the last patient planning duration is less than the OT overtime horizon, we will calculate the overtime cost and compare it with the rescheduling cost. If the overtime cost is less than the rescheduling cost, we continue with overtime. The cost of each patient is different because it depends on the patient type which we will explore later.

3.3 The Model

A patient goes through the surgery process on a fixed day, at a fixed hour which we plan with a daily schedule. The patient $i \in 1, ..., L$ must be scheduled in operation theatre during time horizon, T times units each day. We fix time horizon duration for emergency patients every day. We have one period time and planning horizon is D days. Decision variables are x_{id} (constraint 6) and y_{dj} (constraint 7).

Notation:

i: Patient

j: The index for band of time over time horizon

J: The number of planning time band after time horizon

L: The sum of patient

D: Planning Horizon Day

d: Day

$\overline{T_i}$: The tardiness for patient i

T: Time Horizon

t_i: The surgical duration for patient i

f_j: Extra time allowed

T_d: The sum for regularly time availability of operation room for day d in minutes

C_i: The cost for patient, i

n_i^a: Number of night patient i has spent in the hospital waiting for operation

n_i^h: Number of times patient i has been rescheduled

$C^a_{n_i^a}$: The penalty for a patient spending n_i^a night in the hospital

$C^h_{n_i^h}$: The penalty for a patient being rescheduled n_i^h times

w_d: Penalty day d

Δ_i: Penalty for unit time for not treating the patient within τ_i time units after referred time

v_j: The penalty operation theatre after Time horizon in time section j

r_i: The refer time for each patient i

τ_i: Waiting time limit for patient i

3.3.1 Objective function

The objective is to minimise the delays in treatment and overtime based on the cost for every patient depending on their cases. We have three penalties in this model:

1. Penalty for unit time for not treating the patient within τ time after referred time.
2. The Tardiness for each patient.
3. The Penalty for operation theatre after time horizon.

3.3.2 Constraints

We consider several constraints in our model, and the constraints are given as follow:

1. Specify that the sum of time in OT can only accommodate a limited number of surgical hours per day.
2. Each patient's operation is processed only once. This means that no patients will be processed more than once, and once it has been processed it will no longer be considered in the model.
3. Cost for each patient incurred when the patient's time is booked. This means that if the patient has any costs associated with it at the booked time, it will be included in the model.
4. The penalty of operation room is processed only once. This means that there is no double charging of penalty.
5. The value of tardiness for each patient is positive. This means that the tardiness value will always be positive and no negative value will be considered.

3.3.3 Decision Variables

The decision variable x_{id} is the variable that shows if patient i is treated on day d in time horizon T_d. It will be *1* if it is treated on a particular day d, and *0* if the patient has not been treated. Meanwhile, the decision variable y_{dj} is the variable that shows if the patient is treated on day d in time horizon section j. It will be *1* if the patient is treated on day d in time horizon section j and *0* if it has not been treated. The variables are as follows:

1.
$$x_{id} = \begin{cases} 1 & \text{if surgical patient } i \text{ is treated on day } d \\ 0 & \text{otherwise.} \end{cases}$$

2.
$$y_{dj} = \begin{cases} 1 & \text{if surgical patient is treated on day } d \text{ in time horizon section } j \\ 0 & \text{otherwise.} \end{cases}$$

3.3.4 Mathematical model of the problem(Off-line)

We begin by considering an offline model where full knowledge of the data is known.

$$\text{minimize}(P) = \sum_{i=1}^{L} \Delta_i \overline{T_i} + \sum_{d=1}^{D} \sum_{j=1}^{J} v_j y_{dj} \tag{3.1}$$

subject to:

$$\sum_{i=1}^{L} t_i x_{id} \leq T_d + \sum_{j=1}^{J} f_j y_{dj}, \qquad \forall d = 1, 2, \ldots, D \tag{3.2}$$

$$\sum_{d=1}^{D} x_{id} = 1 \qquad \forall i = 1, 2, \ldots, L \tag{3.3}$$

$$\overline{T_i} \geq \sum_{d=1}^{D} dx_{id} - r_i - \tau_i, \qquad \forall i = 1, 2, \ldots, L \tag{3.4}$$

$$\sum_{j=1}^{J} y_{dj} \leq 1, \qquad \forall d = 1, 2, \ldots, D \tag{3.5}$$

$$x_{id} \in \{0, 1\}, \qquad \forall i = 1, 2, \ldots, L, \quad \forall d = 1, 2, \ldots, D \tag{3.6}$$

$$y_{dj} \in \{0, 1\}, \qquad \forall d = 1, 2, \ldots, D, \quad \forall j = 1, 2, \ldots, J \tag{3.7}$$

The focus is to schedule the patients into the system using the heuristic technique for initial solution. The algorithm is designed to consider patient priority and to achieve the aim of minimum cost. However, the cost for each patient is calculated differently. Its function depends on the types of patients and types of delay.

3.3.5 Types of Patient

We have five types of patients in this problem. Each type carries with it a different type of priority and costs. The order of priority is in decreasing order. As shown in the priority list below, patients with the highest priority are delayed emergency patients. This means that the model will always consider to schedule the delayed emergency patients first, followed by emergency patients. New patients has the lowest priority but once it is included in the system, the priority will change to number 4.

1. Delayed Emergency Patients
2. Emergency Patients
3. Patients Delayed at Hospital
4. Regular Patients Booked and Rebooked
5. New Patients

3.3.6 Cost Patient (On-line)

If all data are known and that the patients list does not change, we do not need to update our model. However, emergency patients arrive online in the sense that once they arrive, they will be included into the system. At the end of the day, these emergency patients need to be scheduled into the OT slots. This will cause the type of patients to change. Since the rescheduling of patients depends on patients' priority, it is important to include this in our model.

We augment $\overline{T_i}$ and define the cost for each patient as:

$$C_i = \left[\Delta_i \, max\left\{\left(T_i - r_i - \tau_i\right), 0\right\} + C^a_{n^a_i} + C^h_{n^h_i}\right] \tag{3.8}$$

where Δ_i is penalty for unit time for not treating the patient within τ_i time units after being referred. The notation τ_i is the waiting time limit for every patient after they are referred into the system. It is more general to allow patient i a waiting time of up to time τ_i without incurring a penalty, than to give the same unpenalised waiting time to each patient. Therefore, the hospital can assign a value of τ_i depending on the type of treatment that patient i needs. T_i is the day on which patient is treated and r_i is referred time for each patient. If T_i is bigger than $r_i + \tau_i$, then we have linear penalty, Δ_i. Otherwise, linear penalty is 0.

We have the penalty for patient delay at home, $C^h_{n^h_i}$ which depends on delay's number, n^h_i and the penalty for patient delay at hospital, $C^a_{n^a_i}$ which depends on delay's number, n^a_i. For $\Delta_i \left(T_i - r_i - \tau_i\right)$ and $C^h_{n^h_i}$ are just for non emergency patients because the cost for emergency patients is zero and the patients go straight into the hospital the soonest possible to get treatment. They cannot be delayed at home and any delay will be delayed at the hospital.

Every day, OT has time horizon less or equal to M minutes. We set different rules depending on the case before the end of the day. We fix N minutes in time horizon for the emergency case every day, and the priority is higher than the other patients. The time N is less than M, $N < M$. We schedule the emergency case as soon as possible to the next day (see Figure 3.1).

Sometimes, planning duration for a patient is different from the actual duration. It might take a longer time. If the sum of duration time is greater than M minutes, the overtime usage of OT will get the penalty depending on the slot time after time horizon.

At the end of the day, if the balance time horizon is less than fifty percent compared to the planned duration time for the next patients, we can decide whether to continue with the surgery or reschedule to another day. If the patient is already delayed at the hospital, we consider performing the surgery but we will get the penalty and the cost for penalty using the OT. This will be compared with the penalty if we delay at the hospital again. Therefore, decisions will be made based on minimum penalty incurred. If the penalty of another delay at the hospital is smaller, other patients must be rescheduled at the end of that day.

3.3.6.1 Overtime Costs

Notation

A_d: The sum of time of OT for day d

t_i^p: The surgical planning time duration of patient i

d: The day on which patient i is treated

c_j: The slot time with the index j after time horizon greater than T

$w_{d'}$: The planning penalty for time horizon greater than T

v_j: The penalty operation theatre after time horizon greater than T

The planning penalty for time horizon greater than T:

$$w_d^{'} = \begin{cases} v_1 & T < A_d + t_i^p \leq c_1 \\ v_2 & c_1 < A_d + t_i^p \leq c_2 \\ v_3 & c_2 < A_d + t_i^p \leq c_3 \\ v_4 & c_3 < A_d + t_i^p \leq c_4 \\ v_5 & c_4 < A_d + t_i^p \leq c_5 \\ v_6 & A_d + t_i^p > c_5 \end{cases}$$

The cost for overtime usage of OT, C:

$$C = \begin{cases} 0 & A_d \leq T \\ v_1 & T < A_d \leq c_1 \\ v_2 & c_1 < A_d \leq c_2 \\ v_3 & c_2 < A_d \leq c_3 \\ v_4 & c_3 < A_d \leq c_4 \\ v_5 & c_4 < A_d \leq c_5 \\ v_6 & A_d > c_5 \end{cases}$$

The planning penalty for time horizon greater than T, $w_d^{'}$ is the penalty in the planning stage when it is decided that the surgery will be performed during overtime usage of OT. However, since the surgery might be shorter or longer than planned, the cost for overtime usage of OT, C is the actual cost calculated after the surgery has finished. That is why the two variables show the same value. One is calculated using the planning duration while the other is calculated using the actual duration.

We need to check the cost for patients and their penalty if they are delayed again at the hospital. We also need to check the costs of overtime usage of the OT. After both costs have been obtained, we compare and decide whether to treat or reschedule the patients. For example, if the penalty for delay is v_1 and the cost of overtime is v_2, then we decide to let the patient be delayed at the hospital and reschedule the patient again. However, if the opposite occurs, the penalty for delay is v_2 and the cost of overtime is v_1, then we decide to perform the surgery with the overtime. Too much extra time will increase the costs as shown in Figure 3.2.

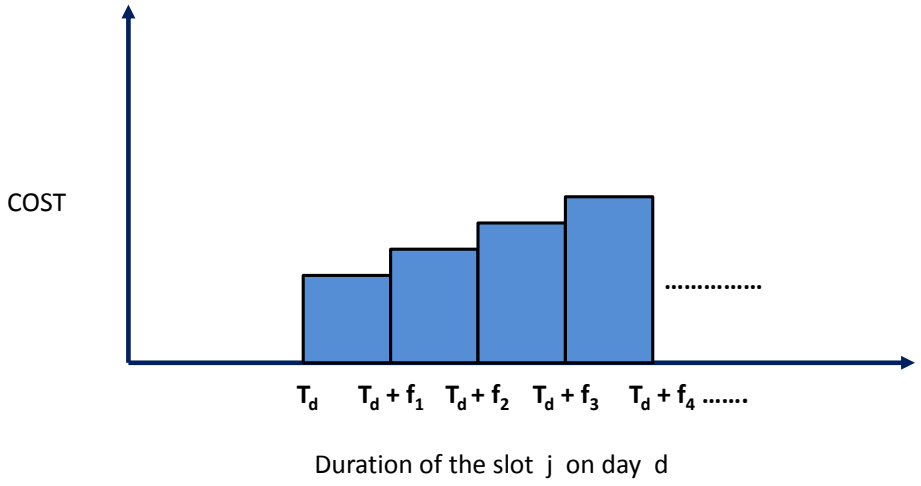

FIGURE 3.2: Duration after time horizon

3.4 Example of Heuristic Technique

As stated in the previous chapter, a heuristic technique (or simply heuristic) is a method which seeks good solutions at a reasonable computation cost. We used

heuristic as a first step in our research because a solution can be obtained with reasonable computational effort. In addition, the solution should be near optimal (with high probability) and the likelihood for obtaining a bad solution (far from optimal) should be low (Martí and Reinelt (2011)). In this section we discuss the heuristic used in our research.

We generate a simple data for 3 days with 23 total number of patients. The waiting time limit for patient after they are referred into the system, τ_i is 5 and penalty for unit time for not treating the patient within the reasonable time is 15. We set time horizon at 6 days in the system.

Again, the focus is on OT Scheduling Problem for emergency patient and regular patient. Patients arrive online and we do not know exactly what type of patients will arrive in the system. Every day, we schedule patient in the empty slot in OT time horizon based on the current information available. The schedule is updated daily to take into account the variations from planned durations and then arrival of emergency patient.

A high number of emergency patient arriving into the system and operation times longer than expected can make other patients being delayed at the hospital or delay at home. At the end of each day, the system comes out with list of schedule and rescheduled patients depending on their urgency.

Table 3.1 shows the list of patient when we generate planning duration which has uniform distribution $T_p \sim [30, 130]$ and the actual duration is the total of T_p and T_{dev}, $T_{dev} \sim [-20, 40]$, with an average of 4.5 regular patients of 1.5 emergency patients every day arriving into the system. However, the type of patient depends on how urgent the patient is. The number of type's patient are 4 for regular patient already in the system, 2 for new emergency patient and 5 for new regular patient.

TABLE 3.1: Simple Data for 3 days

Patient	Refer Date	Planning Duration	Actual Duration	Type of patient
1	-5	106	110	4
2	-5	89	47	4
3	-4	72	74	4
4	-4	113	103	4
5	-4	109	92	4
6	-3	106	106	4
7	-2	104	119	4
8	-2	112	53	4
9	-1	73	81	4
10	-1	106	125	4
11	1	41	66	2
12	1	110	123	5
13	1	76	104	5
14	1	46	68	5
15	1	95	85	5
16	2	113	110	5
17	2	78	63	5
18	2	64	60	5
19	3	67	59	2
20	3	89	84	2
21	3	69	65	2
22	3	46	67	2
23	3	40	11	5

Tables 3.2 and 3.3 shows the list of scheduled and rescheduled patients in the system. The system has a list of initial solution with previously referred patients. We schedule the patients using First Fit Bin Packing Strategy in the OT slots. This means that the algorithm attempts to place the patients in the OT with the first day that can accommodate the patient. If no day is found, it schedule on a new day and puts the patients within the new day. Each day, we apply Daily Scheduling Algorithm to treat the patient on that day, and calculate the costs for each patient and calculate the number of delays if it has happened.

If it seems overtime will be needed, we compare the cost of overtime using OT with the cost for patient and their penalty for delay at hospital to decide whether to treat or reschedule this patient. Lastly, we introduce new emergency patients

TABLE 3.2: List of Initial Schedule

Day	Patient	Planning Duration	Refer Date	Priority	Type of patient
1	4	113	-4	4	4
1	8	112	-2	4	4
1	5	109	-4	4	4
2	1	106	-5	4	4
2	6	106	-3	4	4
2	10	106	-1	4	4
3	7	104	-2	4	4
3	2	89	-5	4	4
3	9	73	-1	4	4
3	3	72	-4	4	4
0	11	41	1	0	2
0	12	110	1	0	5
0	13	76	1	0	5
0	14	46	1	0	5
0	15	95	1	0	5
0	16	113	2	0	5
0	17	78	2	0	5
0	18	64	2	0	5
0	19	67	3	0	2
0	20	89	3	0	2
0	21	69	3	0	2
0	22	46	3	0	2
0	23	40	3	0	5

and new regular patients into the system at the end of this day and schedule into the system using heuristic technique into the system.

We schedule or reschedule patient depending on the patient priority. The priority value is the value used to represent the type of patients as stated in section 3.3.5. This means that the model will always consider to schedule the delayed emergency patients first and is follow by emergency patients. The priority list for important patient such as delayed emergency and emergency patient are 1 and 2. We try to avoid moving those already booked on that day such as those delayed several times at the hospital (type 3) and patients delayed several times at home (type 4).

Lastly we have new patients (type 5). In this example, we fix the empty

TABLE 3.3: List of patient in the system using heuristic technique

Day	Patient	Refer Date	Type of Patient	Planning Duration	Actual Duration	Delay at Hospital	Delay at Home	Cost
1	4	-4	4	113	103	0	0	0
1	8	-2	4	112	53	0	0	0
1	5	-4	4	109	92	0	0	0
2	11	1	2	41	66	0	0	0
2	1	-5	4	106	110	0	0	30
2	6	-3	4	106	106	0	0	0
2	10	-1	4	106	125	0	0	0
3	7	-2	4	104	119	0	0	0
3	2	-5	4	89	47	0	0	45
3	9	-1	4	73	81	0	0	0
3	3	-4	4	72	74	0	0	30
4	22	3	2	46	67	0	0	0
4	19	3	2	67	59	0	0	0
4	20	3	2	89	84	0	0	0
4	21	3	2	69	65	0	0	0
4	15	1	4	95	85	0	0	0
4	12	1	4	110	123	0	0	0
5	18	2	4	64	60	0	0	0
5	13	1	4	76	104	0	1	10
5	14	1	4	46	68	0	1	10
5	16	2	4	113	110	0	0	0
5	17	2	4	78	63	0	0	0
6	23	3	4	40	11	0	0	0

time slot for emergency patient in one day at 120 minutes and schedule emergency patient to the next day or the worse is two days after they come to the hospital. 120 minutes is chosen because since we expect 1.5 emergency patients to arrive, and the mean of the planning duration that follows a uniform distribution $U \sim [30, 130]$ is 80, hence the amount allocated is $1.5 \times 80 = 120$. Moreover, Wullink et al. (2007) showed that the policy of reserving capacity for emergency surgery in all elective OTs led to an improvement in waiting times for emergency.

In this example, Table 3.3 shows patient 13 and 14 are delayed at home and their cost are 10 for each. On day 4, all the emergency patients (patient 19, 20, 21, and 22) come to the hospital on day 3, we schedule into the system on the

next day and reschedule regular patients treatment on day 4, 5 and 6.

The result achieved with our objective function is related with cost of OT, cost when emergency patients are delayed at hospital, cost when patients are delayed at hospital, cost when patients are delayed at home and cost when patients are not treated within the reasonable time after they come to the system are presented in Table 3.4.

TABLE 3.4: Cost of Operation in System OT

Total patient	23
Total cost	175
Cost of OT	50
Cost delay of emergency patient	0
Cost delay at hospital	0
Cost delay at home	20

3.4.1 Example of Manual Calculation

We will show some example in calculating the cost patient that is shown in Table 3.3. With reference to Equation 3.8, we set $\Delta_i = 15$ and $\tau_i = 5$ for all patients i.

Regular Patient :

(i) Patient 4:

$$\begin{aligned} C_4 &= 15 \times \ max\{(1-(-4)-5), 0\} \\ &= 15 \times \ max\{0, 0\} \\ &= 0 \end{aligned} \tag{3.9}$$

(ii) Patient 1:

$$\begin{aligned} C_1 &= 15 \times \ max\{(2-(-5)-5),0\} \\ &= 15 \times \ max\{2,0\} \\ &= 15 \times 2 = 30 \end{aligned} \tag{3.10}$$

Patient delayed at home :

(i) Patient 13: we set $C^h_{n^1_{13}} = 10$

$$\begin{aligned} C_{13} &= [15 \times \ max\{(5-(1)-5),0\}] + 10 \\ &= [15 \times \ max\{-1,0\}] + 10 \\ &= [15 \times 0] + 10 = 10 \end{aligned} \tag{3.11}$$

3.5 Online Procedure

The aim of Online Schedule Planning Procedure is to update the existing schedule every day. We have an initial solution to start our heuristic technique. Each day, a new schedule is created based on current information. The schedule is updated daily to take into account the variations from planned durations and the arrival of emergency patients. We allocate the patients depending on their urgency using a heuristic method.

Algorithm 3.1 Online Procedure

1: Apply algorithm 3.2 to get initial schedule of patients
2: Initialize by setting the day counter as $d = 1$
3: Apply algorithm 3.3 to treat the patient on day d
4: Apply algorithm 3.4 to reschedule patient delay, schedule emergency and new patient
5: Termination Test: If $d <$ Planning Horizon Day, D, set $d = d + 1$ and go to 3

Under the online procedure in algorithm 3.1, we initialise the system with previously referred patients. Algorithm 3.1 is the main algorithm because it incorporates several elements. Firstly, we need to get an initial solution based on the current patient list and this is shown in algorithm 3.2.

When we have an initial solution, we then proceed to the second step of algorithm 3.2 where we set the day = 1, the first day of our new schedule. Then we go to step three and apply algorithm 3.3 to schedule the patient on day d and then proceed to algorithm 3.4 to reschedule the patients with the arrival of emergency patients, new patients and any patients that have been delayed. Step five is basically a termination procedure where the algorithm will loop itself to step three (with $d = d+1$) as long as the Planning Horizon Day, D is not reached.

Before we present the algorithm for the next step, this is the strategy to create the list of patients in initial solution. For easy reference in each of the algorithm, we define the following notations :

Notation:

$S1$: Delayed Emergency Patients

$S2$: Emergency Patients

$S3$: Patients Delayed at Hospital

$S4$: Regular Patients Booked and Rebooked

$S5$: Regular Patients

A_d: The sum of time of OT for day d

t_i^p: The surgical planning time duration of patient i

t_i: The surgical duration of patient i

j: The index for band of time time horizon

d: The day d, the patient i is treated

d^*: The following days of planning after the current day

c_j: The slot time with the index j after time horizon greater than T

b_i: The day patient i goes to hospital

r_i: The refer time for each patient

n_i^a: Number of night patient i has spent in the hospital waiting for operation

n_i^h: Number of times patient i has been rescheduled

T_i: The day on which patient i is treated

C: Cost for overtime usage of OT

D: Planning horizon day

T^e: Fix time for emergency case in horizon day everyday

3.5.1 Algorithm Initial Scheduling

Algorithm 3.2 Initial Solution

1: Consider $S5$ by referring the data order. Only $S5$ are considered since this is an initial schedule and the patients are referred to their referral date.

2: Schedule $S5$ to the available time slots using First Fit Bin Packing Strategy. We schedule patients, $S5$ into the OT with patient that has the longest planned duration first and followed by other $S5$ in descending order. We schedule until all slots are filled in time horizon every day. If the slots are full on that day, we schedule on the next day until all $S5$ are scheduled into the system.

3: We set number of delayed at home, n_i^h and delayed when admitted to hospital, n_i^a to zero

Under algorithm 3.2 we need to consider all the regular patients in the system. We then schedule these patients using the First Fit Bin Packing Strategy. In general, our problem is similar to the Dual Bin Packing Problem since we are given a fixed number of OT with a fixed available usage time. Boyar and Favrholdt (2003) stated that for Dual Bin Packing Problem, First Fit is better than Best

Fit, which is better than Worst Fit. In scheduling the regular patients, we use the strategy commonly known as longest serving time first where patients with the longest planned duration will be scheduled first during the day to minimise the possibility of overtime. When all the slots in the time horizon has been filled, the remaining patients will be scheduled on the next day. Since this is an initial solution, we set the number of patients delayed at home and at hospital at zero.

3.5.2 Algorithm Daily Schedule

In this Daily Schedule algorithm, we evaluate at day d. We treat patients in the system in time horizon at day d for patients $S1$,or $S2$ or $S3$ or $S4$. Therefore, at the end of day d, we will plan $S1$ and $S3$ to join the system and new patients, $S5$ and new emergency patients, $S2$ will be introduced into the system and they will be schedules with reschedule algorithm. We also calculate the cost of a patient in the system and how many times a patient is delayed if it happens. However, for some cases the operation times is longer than expected and will need overtime usage of OT. If this happen, we calculate the sum of time of OT for the day including half of the planned time for the last patient of the day. If it is less than the time horizon, we check the cost for patients and their penalty for delay at hospital. Comparison will then be made with the cost of overtime usage of OT to decide whether to treat at that day or to reschedule the patient to another day.

Algorithm 3.3 Daily Schedule

1: Form a list L of patients that contains $S1$ in order from the longest to shortest duration and gradually $S2$, $S3$ and $S4$ with $T_i = d$ also ordered according to arrival time and τ, then longest to shortest duration
Set $A_d = 0$

2: Choose the next patient, i in list L. If there is no next patient in L, go to 7

3: If $A_d + t_i^p \leq T$, $A_d = A_d + t_i$
If $i \in S1$ or $i \in S2$ or $i \in S3$, $n_i^a = d - b_i$
Calculate $C_i = \left[\Delta_i \, max\left\{\left(T_i - r_i - \tau\right), 0\right\} + C^a_{n_i^a} + C^h_{n_i^h}\right]$
Remove i from system and go to 2

4: If $A_d + t_i^p/2 \leq T$, $A_d = A_d + t_i$
If $i \in S1$ or $i \in S2$ or $i \in S3$, $n_i^a = d - b_i$
Calculate $C_i = \left[\Delta_i \, max\left\{\left(T_i - r_i - \tau\right), 0\right\} + C^a_{n_i^a} + C^h_{n_i^h}\right]$
Remove i from system and go to 2

5: If $n_i^a \geq 1$
If $w_d^{'} - w_d \leq C^a_{n^a_{i+1}} - C^a_{n_i^a}$
$A_d = A_d + t_i$
If $i \in S1$ or $i \in S2$ or $i \in S3$, $n_i^a = d - b_i$
Calculate $C_i = \left[\Delta_i \, max\left\{\left(T_i - r_i - \tau\right), 0\right\} + C^a_{n_i^a} + C^h_{n_i^h}\right]$
Remove i from system and go to 2
else $n_i^a = n_i^a + 1$
If $i \in S2$ then set $S1 = S1 \cup \{i\}$, set $T_i = 0$, $S2 = S2 \setminus \{i\}$ and go to 2
If $i \in S4$ then set $T_i = 0$, $S3 = S3 \cup \{i\}$, $S4 = S4 \setminus \{i\}$, if $b_i = 0$
set $b_i = d$ and go to 2

6: else $n_i^a = n_i^a + 1$
If $i \in S2$ then set $S1 = S1 \cup \{i\}$, set $T_i = 0$, $S2 = S2 \setminus \{i\}$ and go to 2
If $i \in S4$ then set $T_i = 0$, $S3 = S3 \cup \{i\}$, $S4 = S4 \setminus \{i\}$, if $b_i = 0$ set $b_i = d$ and go to 2
else go to 2

7: Calculate C, Cost for overtime using OT

8: For each new regular patient and is not an emergency patient,i, set $S5 = S5 \cup \{i\}$,
$r_i = d$ and $T_i = 0$
For each new emergency patient,i set $S2 = S2 \cup \{i\}$, $r_i = d + 1$, $T_i = 0$ and $b_i = d + 1$
Set $n_i^h = n_i^a = 0$

9: Stop

3.5.3 Algorithm Rescheduling

From the Daily Schedule algorithm, we will have patients leaving the system but there will be patients that are delayed such as delayed emergency patients and delayed at the hospital, patients $S1$ and $S3$. Furthermore, emergency patients ($S2$) and new patients ($S5$) will also be introduced into the system. We schedule new $S2$ and new $S5$ that we plan at day $d+1$. If delay occurs, we reschedule $S1$ and $S3$ into the system.

We design algorithm, with the aim to minimise the cost of a patient depending on the priority of the patient . The idea of to prioritise is so that more important patients such as delayed emergency and emergency patients can be treated at the beginning of the day. Therefore, higher priority will be given to patients delayed many times at home or hospitals, compared to the first-time delayed patients. So less important patients can rescheduled into next day. After rearranging the schedules, other patients in the system that we plan at day $d+1$ might be delayed ($S1$ or $S4$). We will avoid moving patients that are already booked and patients that have been delayed many time at the hospital. We calculate how many times patients are delayed if it happens and calculate the total cost for the system. Again, we use the first bin packing method in this algorithm.

The rescheduling algorithm is efficient in the sense that it will schedule the patients that need treatment most first and try to avoid moving those already booked because it will cause more delays which will increase the cost. Since it takes into account the urgency that is attached with patient priority, it will eliminate the possibility that urgent patients do not receive the appropriate treatment immediately.

Algorithm 3.4 Reschedule

1: Set a day, $d = d^*$

2: Form again a list L of patient that contains $S1$ ordered depends on n_i^a then by waiting time,
$S2$ is ordered how long by waiting time,
$S3$ with $T_i = 0$ ordered depends on n_i^a then by arrival time and τ from the refer time,
$S4$ with $T_i = d$ ordered by longest to shortest duration
and $S4$ with $T_i = 0$ ordered depends on n_i^h then longest and shortest duration and
$S5$, $T_i = 0$ also ordered according by arrival time and τ, then longest to shortest duration
Set $A_d^p = 0$

3: If L fully searched goto 7
else choose next patient i in list L

4: If $i \in S1$ or $i \in S2$, or $d = d^*$
If $A_d^p + t_i^p \leq T$, $A_d^p = A_d^p + t_i^p$, set $T_i = d$ remove i from L and goto 5
else goto 6
else If $A_d^p + t_i^p \leq T - T^e$, $A_d^p = A_d^p + t_i^p$, set $T_i = d$ remove i from L and goto 5
else goto 6

5: If $i \in S5$ then set $S4 = S4 \cup \{i\}$, $S5 = S5 \setminus \{i\}$, End If
goto 3

6: If $i \in S4$, with $T_i = d$, then set $T_i = 0$, $n_i^h = n_i^h + 1$, End If
goto 3

7: If $d < D$, set $d = d + 1$ and go to 2
At the end of horizon, treat all remain patients on day $D + 1$

8: Stop

3.6 Local Search Method

Local search method approach to hard combinatorial optimization problem is discussed by Burke and Kendall (2005). The basic idea of the local search method is to consider the neighbour of the current solution and try to locate a better solution. If we are able to locate a better solution, we then explore the neighbour

of this particular solution. We iterate this procedure until no better solution can be located and hence the current solution is accepted as the best solution.

One of the important issues when implementing a local search procedure is how to pick initial solution as well as how to define neighbourhood and to select neighbour of a given solution. This is particularly important because we might get an optimal solution but only a local optima and not a global optima. A local optima is a solution that is optimal within a neighbouring set of candidate solutions. On the other hand, a global optima is the optimal solution among all possible solutions. Crama et al. (1995) suggested that in many cases, finding an initial solution is not difficult but the choice of this starting solution may greatly affect the quality of the final outcome.

Some notations to be used in our local search method are defined below.

Notation:

C: Cost for overtime using Operation Theatre

d_i: The current day book

d_i': The day booked that patient's knows(new and emergency patient are zero)

T_j: The current day for consider to treat it

b_i: Patient come to the hospital

n_i^h: Number of times patient i has been rescheduled

$n_i^{h\,\prime}$: The total number of times patient i has been rescheduled and the patient knows their day's booked

r_i: The refer time for each patient

Δ_i: Penalty for unit time for not treating the patient within τ time units after referred

3.6.1 Cost For Local Search

We define a cost formula for the local search algorithm. The new cost for penalty delay at home is different compared to the cost of the patient in the model at section 3.3.6. The new cost now depends on δ_i, the variable that represents the rescheduling decision of the patient. The variable is defined as follows:

$$C = \Delta_i \, max \, \{(T_j - r_i - \tau, 0)\} + C^a_{T_j - b_i} + C^h_{n_i^h + \delta_i}$$

$$C^h_{n_i^h}, \; n_i^h = n_i^{h'} + \delta_i,$$

$$\delta_i = \begin{cases} 0 & d'_i = 0 \\ 1 & d'_i \neq 0 \qquad \& \qquad T_j \neq d'_i \\ 0 & otherwise \end{cases}$$

$$C^a_{T_j - b_i} = \begin{cases} C^a_{T_j - b_i} & T_j - b_i > 0 \\ 0 & otherwise \end{cases}$$

From the cost equation C above, we can see that the penalty cost for delayed at home depends on δ_i where it is equal to one when the patients knows when they are scheduled. However, when we apply the local search algorithm, the new scheduled day is not the same as the day known to the patients. This means that the patients will be delayed again and hence the number of times patient i has been rescheduled will increase by 1. Once a patient enters the system, a day is assigned where the surgery of the patient will be performed. If the day is informed to the patient then the patient is considered to "know" their surgery date. If the date changes to a later date, then the patient is considered to have been rescheduled since they now have to wait longer. Every time a date changes, a cost is incurred. If the date changes several times, more cost is incurred. However, if the patient is not informed about the date, then any changes to the date will

not affect rescheduling cost. If the patients do not know when they are scheduled, any changes in the schedule will not cause any delay.

Another factor that has changed is the penalty cost for patients that has been delayed at the hospital. When we apply the local search algorithm, some patients might be scheduled earlier hence no delay will occur and the cost is zero. If the new schedule date is later than the day the patients arrive at the hospital, then delay will occur and hence the delay cost will be incurred depending on how long the patients is delayed.

3.6.2 Algorithm Local Search

Under the Local Search algorithm, we start with the heuristic technique as an initial solution but we modify the reschedule algorithm (3.4) with an extra step where we apply the local search algorithm after step 7. Following the previous algorithms, at the end of each day, we schedule new patients into the system and reschedule patients delay at hospital and home if any delay happens, where we use heuristic technique as an initial solution.

Now, under the local search algorithm (3.5), we begin by swapping every pair of patients if they satisfied the condition that we imposed. For example, we do not swap patients on the same day. Any emergency patients can only be swapped into the next day. Before we swap the patients, we need to check the sum of time on the duration day. If emergency patients are swapped into the new day, the sum of duration time must be less than or equal to T. If regular patients are swapped, the sum of duration time must be less than or equal to $T - T^e$. If this condition is satisfied, the pair of patients will be swapped.

After the patients are swapped, we check the total cost of the swap and compare it with the current cost. If the new total cost is less than the current

cost, the swap will be finalised. We then consider the next patient (going through the same procedures) until all the balance patients in the list have been considered and we come out with the new list of schedule for the next day.

Notation:

$A^p_{d_{\pi(i)}}$: The sum of planning time booked of operation room for day $d_{\pi(i)}$

T: time horizon

T_i: The operation time for a patient i

T_e: Fix time for emergency case in horizon day everyday

I: The total number of patient's in order list

INPUT:

d^*: The following days of planning after the current day

$d_{\pi(i)}$: The day on which patient $\pi(i)$ is treated

$t^p_{\pi(i)}$: The surgical planning time duration of patient $\pi(i)$

$b_{\pi(i)}$: The day patient $\pi(i)$ goes to hospital

$\pi(i)$: Patient that correspond to position i

i: Position of patient i in the list

i^*: Position of patient i after swap

j: Position of patient j in the list

j^*: Position of patient j after swap

Algorithm 3.5 Local Search

0: Set $i^* = I - 1$ and $j^* = I$

1: Set $i = 1$ and $j = 2$

2: If $d_{\pi(i)} = d^*$

a) For emergency patients:

while $d_{\pi(j)} = d_{\pi(i)}$, set $j = j + 1$

if $j > I$, go to 5

if$(A^p_{d_{\pi(i)}} - t^p_{\pi(i)} + t^p_{\pi(j)} \leq T)$ and $(A^p_{d_{\pi(j)}} - t^p_{\pi(j)} + t^p_{\pi(i)} \leq T - T^e)$, go to 4

else go to 5

b) For regular patients:

while $d_{\pi(j)} = d_{\pi(i)}$, set $j = j + 1$

if $j > I$, go to 5

if$(A^p_{d_{\pi(i)}} - t^p_{\pi(i)} + t^p_{\pi(j)} \leq T - T^e)$ and $(A^p_{d_{\pi(j)}} - t^p_{\pi(j)} + t^p_{\pi(i)} \leq T - T^e)$, go to 4

else go to 5

3: Calculate $TotalNewCost(i + j)$ for $T_i + T_j$

if $TotalNewCost(i + j) \geq TotalCurrentCost(i + j)$, go to 5

else $TotalCurrentCost(i + j) = TotalNewCost(i + j)$

$Dummy = \pi(i)$

$\pi(i) = \pi(j)$

$\pi(j) = Dummy$

set $i^* = i$ and $j^* = j$

end if

set $j = j + 1$

if $j \leq I$, go to 2

else set $i = i + 1$ and $j = i + 1$

if $i < I$, go to 2

else go to 1

4: Set $j = j + 1$

if $i^* = i$ and $j^* = j$, stop

if $j \leq I$, go to 2

else set $i = i + 1$ and $j = i + 1$

if $i^* = i$ and $j^* = j$, stop

if $i < I$, go to 2

else go to 1

3.7 Simulated Annealing Method

The concepts of Simulated Annealing in Combinatorial Optimization were first introduced independently by Kirkpatrick et al. (1983) and Cerny (1985) in the early 1980s, and a thorough discussion can be found in Burke and Kendall (2005). The idea is the thermal process to obtain low energy states of solids in a heat bath. Kirkpatrick et al. (1983) discussed the two steps in the thermal process. First, we increase the temperature of the heat bath to maximum value until the solid melts and second, we decrease carefully the temperature of the heat bath until the particles arrange themselves in the ground state of the solid.

In our research, we calculate the difference of the total cost (total new cost − total current cost), Δ between a pair of patients that we plan to swap. The difference between this procedure and the local search procedure is that even when the difference of the total cost is positive, the swapping might still happen but only with a certain probability.

If Δ is less than or equal to zero, we swap the patients but if Δ is more than zero, we swap with probability $e^{-\Delta/K}$, where K is the temperature and we have a geometric cooling $K = \alpha K$. The value used for α is typically about 0.9 but this value depends on us and different value of α can be used such as 0.95 by Sier et al. (1997) where they also use the simulated annealing procedure in the scheduling surgical procedures.

Under this procedure , firstly we must obtain the initial temperature for the process by calculating 20% from the initial swap. The initial swap is obtained by using the previous local search method. After we calculate 20% from the initial swap, we check what is the Δ maximum. By rearranging the probability $e^{-\Delta/K}$, we get the value of the initial temperature, $K = -\Delta/ln(0.2)$. The decision to swap the patients will be done by comparing the probability obtained with each

new value of K and Δ with a random number $r \in [0, 1]$. If r is less than or equal to the probability obtained, the patients are swapped but if r is more than the probability obtained, the patients are not swapped.

The swapping pair of patients procedure will continue with smaller K because of the geometric cooling and will stop when we reach the last temperature K that we fix. The final temperature will be fix according to the initial temperature such that, the cooling process will continue normally until the last temperature according to the criteria that we fixed. At the end of procedure, we will come out with a new list of scheduled patient that achieve our the objectives of our research.

3.7.1 Algorithm Simulated Annealing

We present two versions of simulated annealing algorithm. One is a traditional algorithm that randomly search the solution space to find a better solution in the neighbourhood until the algorithm terminates at a minimum. Under the traditional algorithm, a pair of patients is randomly chosen to be swap. We then updated the algorithm where instead of choosing a pair of patient at random, we systematically loops through the pairs of patients in the list. The notation and symbols used are as follow:

Notation:

A_d^p:	The sum of planning time booked of operation room for day d
T^e:	Fix time for emergency case in horizon day everyday
T_i	The operation time for a patient i
I	The total number of patient's in order list
K_{pr}	Temperature
K'_{pr}	0 or 1 measure of whether cooling is still ongoing

Γ Repetition counter

INPUT:

d^*: The following days of planning after the current day

$d_{\pi(i)}$: The day on which patient $\pi(i)$ is treated

$t^p_{\pi(i)}$: The surgical planning time duration of patient $\pi(i)$

$b_{\pi(i)}$: The day patient $\pi(i)$ goes to hospital

$\pi(i)$: Patient that correspond to position i

i: Position of patient i in the list

i^*: Position of patient i after swap

In the traditional algorithm 3.6, firstly (step 0) we set the list of patients, the temperature to the initial temperature ($K_{pr} = InitialK_{pr}$), and $K'_{pr} = 1$ to indicate the cooling has started. Next in step 1, we choose a pair of patient at random and calculate the new temperature $K_{pr} = \alpha K_{pr}$. The cooling will stop once the temperature is less than the final temperature ($K_{pr} < FinalK_{pr}$) and which point we will set $K'_{pr} = 0$ and the process is stopped. In step 2, we set the repetition counter $\Gamma = 0$.

In step 3, we check to see if the pair of patient is schedule on the same day and if so we will choose a different patient j at random that is not schedule on the same day as patient i. Then we check to see if the patients are compatible to be swap where we check if they will fit in the time horizon for the day. If the patients are compatible to be swapped, in step 4 we calculate Δ, the difference between the new cost of swapping the patients with the current cost. If Δ is less than or equal than zero, we swap the patients but if Δ is more than zero, we calculate the probability $e^{-\Delta/K_{pr}}$ and compare with a random number $r \in [0, 1]$ where the

Algorithm 3.6 Traditional Simulating Annealing

0: Set a list with I patients, $T_{pr} = InitialK_{pr}$ and $K'_{pr} = 1$
1: Choose random i and j, where $i \neq j$
Calculate $K_{pr} = \alpha K_{pr}$, If $K_{pr} < FinalK_{pr}$ set $K'_{pr} = 0$
2: Set repetition counter $\Gamma = 0$
3: While $d_{\pi(j)} = d_{\pi(i)}$, choose a random j where $d_{\pi(i)} \neq d_{\pi(j)}$
if$(A^p_{d_{\pi(i)}} - t^p_{\pi(i)} + t^p_{\pi(j)} \leq T)$ and $(A^p_{d_{\pi(j)}} - t^p_{\pi(j)} + t^p_{\pi(i)} \leq T)$, go to 4
else go to 5
4: Calculate $TotalNewCost(i+j)$ for $T_i + T_j$
Calculate $Delta = TotalNewCost(i+j) - TotalCurrentCost(i+j)$
if $Delta \leq 0$
$TotalCurrentCost(i+j) = TotalNewCost(i+j)$
$Dummy = \pi(i)$
$\pi(i) = \pi(j)$
$\pi(j) = Dummy$
else if $Delta > 0$ and $K'_{pr} = 1$
Calculate $Pro = e^{-\Delta/K_{pr}}$
Derive random number $r \in [0, 1]$
if $r \leq Pro$
$TotalCurrentCost(i+j) = TotalNewCost(i+j)$
$Dummy = \pi(i)$
$\pi(i) = \pi(j)$
$\pi(j) = Dummy$
else go to 5
end if
else choose random i and j, where $i \neq j$
if $\Gamma \leq 20$
update $\Gamma = \Gamma + 1$, go to 3
else go to 1
5: if $K'_{pr} = 0$, stop
else choose a random j
if $\Gamma \leq 20$
update $\Gamma = \Gamma + 1$, go to 3
else go to 1

patients will swapped only if $r \leq e^{-\Delta/K_{pr}}$. Otherwise we choose a different patient j to be swapped and go back to step 3 while the repetition counter is ongoing.

In step 5, the algorithm is stopped once the cooling has stopped. Otherwise, patients will be test to be swapped until the repetition counter is satisfied at which point the algorithm goes back to step 1 where the temperature is gradually lowered until it reaches the final temperature.

In the updated algorithm 3.7, firstly (step 0) we set the list of patients,

the temperature to the initial temperature ($K_{pr} = InitialK_{pr}$), and $K'_{pr} = 1$ to indicate the cooling has started. Next in step 1, we choose the first and second patients in the list and calculate the new temperature $K_{pr} = \alpha K_{pr}$. As in the traditional algorithm, the cooling will stop once the temperature is less than the final temperature ($K_{pr} < FinalK_{pr}$) and which point we will set $K'_{pr} = 0$ and the process is stopped. In step 2 and 3, we check to see if the patients is schedule on the same day, and if they are, the second patient is changed to the next patient in the list until both the patients are not schedule in the same day. Then we check to see if the patients are compatible to be swap where we check if they will fit in the time horizon for the day. If the patients are compatible to be swapped, in step 4 we calculate Δ, the difference between the new cost of swapping the patients with the current cost. If Δ is less than or equal than zero, we swap the patients but if Δ is more than zero, we calculate the probability $e^{-\Delta/K_{pr}}$ and compare with a random number $r \in [0, 1]$ where the patients will swapped only if $r \leq e^{-\Delta/K_{pr}}$. Otherwise we choose the next patient in the list to be patient j to be swapped and go back to step 2. If there is no other patient in the list, we change the patient i to the next patient in the list and choose the next patient after that as patient j and go back to step 2. Once we have gone through the list, we go back to step 1 to choose patient i and j.

In step 5, we check the current temperature and the algorithm is stopped once the cooling has stopped. Otherwise, the algorithm goes back to step 1 where the temperature is gradually lowered until it reaches the final temperature.

Algorithm 3.7 Updated Simulating Annealing

0: Set $i^* = I - 1$, $j^* = I$,
$k_{pr} = InitialK_{pr}$ and $K'_{pr} = 1$
1: Set $i = 1$, $j = 2$
Calculate $K_{pr} = \alpha K_{pr}$, If $K_{pr} < FinalK_{pr}$ set $K'_{pr} = 0$
2: If $d_{\pi(i)} = d^*$
while $d_{\pi(j)} = d_{\pi(i)}$ and $j \leq I$, set $j = j + 1$
if $j > I$, go to 5
if$(A^p_{d_{\pi(i)}} - t^p_{\pi(i)} + t^p_{\pi(j)} \leq T)$ and $(A^p_{d_{\pi(j)}} - t^p_{\pi(j)} + t^p_{\pi(i)} \leq T - T^e)$, go to 4
else go to 5
3: While $d_{\pi(j)} = d_{\pi(i)}$ and $j \leq I$, set $j = j + 1$
if $j > I$, go to 5
if$(A^p_{d_{\pi(i)}} - t^p_{\pi(i)} + t^p_{\pi(j)} \leq T - T^e)$ and $(A^p_{d_{\pi(j)}} - t^p_{\pi(j)} + t^p_{\pi(i)} \leq T - T^e)$, go to 4
else go to 5
4: Calculate $TotalNewCost(i + j)$ for $T_i + T_j$
Calculate $Delta = TotalNewCost(i + j) - TotalCurrentCost(i + j)$
if $Delta \leq 0$
$TotalCurrentCost(i + j) = TotalNewCost(i + j)$
$Dummy = \pi(i)$
$\pi(i) = \pi(j)$
$\pi(j) = Dummy$
set $i^* = i$ and $j^* = j$
else if $Delta > 0$ and $K'_{pr} = 1$
Calculate $Pro = e^{-Delta/K_{pr}}$
Derive random number $r \in [0, 1]$
if $r \leq Pro$
$TotalCurrentCost(i + j) = TotalNewCost(i + j)$
$Dummy = \pi(i)$
$\pi(i) = \pi(j)$
$\pi(j) = Dummy$
set $i^* = i$ and $j^* = j$
else go to 5
end if
set $j = j + 1$
if $j \leq I$, go to 2
else set $i = i + 1$ and $j = i + 1$
if $i < I$, go to 2
else go to 1
5: if $i^* = i$, $j^* = j$ and $K'_{pr} = 0$, stop
else set $j = j + 1$
if $i^* = i$, $j^* = j$ and $K'_{pr} = 0$, stop
if $j \leq I$, go to 2
else set $i = i + 1$ and $j = i + 1$
if $i^* = i$, $j^* = j$ and $K'_{pr} = 0$, stop
if $i < I$, go to 2
else go to 1

3.8 Experimental Design

The main reason why we use generated data and not real data is because we want to see how applicable our method is. By using generated data, we can create different situations that might occur in different hospitals to get an idea whether our algorithm will provide a schedule that achieve the stated aims and objectives. We want to see if one of the algorithm outperforms the other whether in some or all the data sets.

We can check the validity of our method by using different variation of data. We are able to check the range of our data by covering the type of fine that are imposed at some hospitals. We will be able to simulate the condition of which patients arrive and how the decision on how to treat the patients are made (online) between different hospitals because the rate of patients arriving into the system vary every time in real life.

By changing the data, we will be able to look at how the algorithm is applied to real hospital. The hospital can provide their real data and then compare the results of our different algorithm to their current scheduling policy.

3.8.1 Example using Generated Data

We generate a variety of data using Uniform Distribution. The notation $U \sim [a, b]$ denotes that the data are randomly generated from a uniform distribution defined on the interval $[a, b]$. The number of patients arriving each day has the sum of the probability distribution function. The number of regular patients is usually greater than the number of emergency patients.

Let n^r be the number of regular patients arriving on a particular day and n^e be the number of emergency patients arriving on a particular day:

Example 1

Regular Patient: 4.5, Emergency Patient: 1.5, $T_{dev} = [-20, 40]$

$$P(n^r) = \begin{cases} 0.02 & if\ n^r = 0 \\ 0.06 & if\ n^r = 1 \\ 0.10 & if\ n^r = 2 \\ 0.15 & if\ n^r = 3 \\ 0.20 & if\ n^r = 4 \\ 0.20 & if\ n^r = 5 \\ 0.10 & if\ n^r = 6 \\ 0.06 & if\ n^r = 7 \\ 0.05 & if\ n^r = 8 \\ 0.03 & if\ n^r = 9 \\ 0.03 & if\ n^r = 10 \\ 0.00 & if\ n^r > 10 \end{cases}$$

$$P(n^e) = \begin{cases} 0.25 & if\ n^e = 0 \\ 0.30 & if\ n^e = 1 \\ 0.25 & if\ n^e = 2 \\ 0.10 & if\ n^e = 3 \\ 0.10 & if\ n^e = 4 \\ 0.00 & if\ n^e > 4 \end{cases}$$

Example 2

Regular Patient: 4.0, Emergency Patient: 1.5, $T_{dev} = [-20, 40]$

$$P(n^r) = \begin{cases} 0.04 & if\ n^r = 0 \\ 0.12 & if\ n^r = 1 \\ 0.14 & if\ n^r = 2 \\ 0.16 & if\ n^r = 3 \\ 0.16 & if\ n^r = 4 \\ 0.12 & if\ n^r = 5 \\ 0.10 & if\ n^r = 6 \\ 0.07 & if\ n^r = 7 \\ 0.04 & if\ n^r = 8 \\ 0.03 & if\ n^r = 9 \\ 0.02 & if\ n^r = 10 \\ 0.00 & if\ n^r > 10 \end{cases}$$

$$P(n^e) = \begin{cases} 0.25 & if\ n^e = 0 \\ 0.30 & if\ n^e = 1 \\ 0.25 & if\ n^e = 2 \\ 0.10 & if\ n^e = 3 \\ 0.10 & if\ n^e = 4 \\ 0.00 & if\ n^e > 4 \end{cases}$$

Example 3

Regular Patient: 3.5, Emergency Patient: 1.5, $T_{dev} = [-20, 40]$

$$P(n^r) = \begin{cases} 0.04 & if\ n^r = 0 \\ 0.15 & if\ n^r = 1 \\ 0.20 & if\ n^r = 2 \\ 0.18 & if\ n^r = 3 \\ 0.14 & if\ n^r = 4 \\ 0.10 & if\ n^r = 5 \\ 0.08 & if\ n^r = 6 \\ 0.05 & if\ n^r = 7 \\ 0.03 & if\ n^r = 8 \\ 0.02 & if\ n^r = 9 \\ 0.01 & if\ n^r = 10 \\ 0.00 & if\ n^r > 10 \end{cases}$$

$$P(n^e) = \begin{cases} 0.25 & if\ n^e = 0 \\ 0.30 & if\ n^e = 1 \\ 0.25 & if\ n^e = 2 \\ 0.10 & if\ n^e = 3 \\ 0.10 & if\ n^e = 4 \\ 0.00 & if\ n^e > 4 \end{cases}$$

The mean number of regular patients arriving on a particular day is either 4.5, 4.0 or 3.5 (see example 1, 2 and 3) and the mean number emergency patients is 1.5.

The first step is to generate random data $U \sim [0, 1]$ to get the number of patients for emergency cases and regular patients daily. Next, we generate the planning duration time for each patient. The planning duration has uniform distribution $T_p \sim [30, 130]$. We assume the planning duration is between 30 to 130 minutes, with daily means of 6, 5.5 and 5, respectively.

Lastly, we generate the actual duration time. The actual duration is the total of T_p and T_{dev}, $T_{dev} \sim [-20, 40]$. This interval is chosen for T_{dev} because the relative value is small, and the mean is also small. Otherwise the method of estimation for the planning duration is not good if it has a large deviation. We generate random data for one year.

Besides generating data with T_{dev}=[-20,40], we also generate data with T_{dev}=[-25,35]. The results of our methods using the generated data is shown in Table 3.6, Table 3.7, and Table 3.9.

3.9 Testing of Heuristics

In order to see what is the best heuristic technique to used as a sorting method, we run a computational test of several heuristics to see which one performs the best. We calculate the total cost, the delay at home, the delay at hospital and the runtime of the algorithm. We decide to test four heuristics which are:

1. First-Come-First-Serve
2. Longest to Shortest Duration
3. Shortest to Longest Duration
4. Random sorting

The results are shown in Table 3.5. We can see that the sorting method with the lowest total cost is the 'Longest to Shortest Duration' with the total cost of 1523905. Although this method has the highest number of delay at home, at the same time it has a low delay at hospital reducing the total cost. In terms of runtime, all methods have almost the same runtime which suggested that runtime is not affected by the methods.

With this result, we decided to use the 'Longest to Shortest Duration' sorting method for our computational test using different parameter values.

TABLE 3.5: Results of Heuristics Testing

Sorting Method	**Total Cost**	**Delay at Home (Cost/Day)**	**Delay at Hospital (Cost/Day)**	**Runtime (seconds)**
First-Come-First-Serve	1548790	1780/160	2560/32	42
Longest to Shortest Duration	1523905	1910/173	1680/21	46
Shortest to Longest Duration	1589885	1390/137	1520/19	45
Random Sorting	1525305	1570/147	2080/26	46

3.10 Computational Results

We present the computational results of our methods using the generated data. For each average number of patients, we can see that there is no big difference between the number of patients which means that our data sets will be valid for comparison. For easy comparison, we present several graphs comparing the total cost for OT process in the system, cost of overtime for OT, cost for patients delay at hospital, cost patients delay at home, average capacity utilization up to horizon day and the number of patients left after horizon day.

TABLE 3.6: Computational Results using Heuristic Technique

Average patients arrival	Number of patients	Total cost	Cost of delay at Hospital	Cost of delay at home	Cost of OT	Average % between 400 day	day > 400
6	2541	750770	26940	326400	11300	96.4786835	81
6	2552	555185	31795	285960	12700	97.3051987	84
6	2659	760750	46645	350605	15500	97.8781281	107
6	2557	794165	25935	360720	10550	95.8176727	90
6	2519	371475	18585	200385	14100	97.878624	73
5.5	2476	159745	6820	122845	10910	97.0718842	71
5.5	2300	126335	5475	97165	12850	96.3473892	30
5.5	2313	156310	6545	127965	13550	97.4318085	30
5.5	2391	209555	9765	148360	14800	97.0723724	52
5.5	2379	211835	6880	161970	13000	97.3078232	43
5	2157	59995	4510	36560	12850	94.6749725	3
5	2154	43295	6195	16540	12550	96.6572876	3
5	2138	56320	6470	30240	12950	96.4885254	3
5	2125	37415	5475	16225	11950	94.1004868	3
5	2120	35640	4340	13515	13150	94.7203522	3

TABLE 3.7: Computational Results using Local Search Method

Average patient arrival	Number of patient	Total cost	Cost delay at Hospital	Cost delay at home	Cost of Operating Theatre	Average % between 400 day	day > 400
6	2541	732055	25700	298465	10765	96.8909878	79
6	2552	554167	28635	273600	13864	96.6692794	85
6	2659	773520	36510	343800	14456	95.9759308	107
6	2557	438845	26658	336520	11880	96.5193586	89
6	2519	386504	15560	192546	14008	97.2272942	70
5.5	2476	788940	9425	165000	13500	97.1324717	71
5.5	2300	122450	4560	123465	12400	98.7826087	28
5.5	2313	124650	6246	125688	12550	98.7462173	29
5.5	2391	219600	7540	154624	17320	95.9471742	48
5.5	2379	214533	8235	154330	14205	97.9924717	42
5	2157	44620	4436	36050	13840	95.05698423	3
5	2154	44100	6075	16645	12990	95.17745642	3
5	2138	56318	5960	29945	11955	96.33461823	3
5	2125	36695	5290	16200	12100	94.84003935	2
5	2120	35840	4250	13495	11680	94.98768542	2

TABLE 3.8: Computational Results using Traditional Simulated Annealing Method

Average patient arrival	Number of patient	Total cost	Cost delay at Hospital	Cost delay at home	Cost of Operating Theatre	Average % between 400 day	day > 400
6	2541	689824	25500	301690	13650	96.562327	82
6	2552	559760	28355	288540	13520	94.458024	94
6	2659	711450	26500	373750	15388	95.756805	102
6	2557	559884	30824	277685	14486	94.678422	93
6	2519	378550	13382	194486	12558	96.814355	86
5.5	2476	225649	8655	169708	14358	95.894036	71
5.5	2300	159885	5890	120688	10644	96.367132	28
5.5	2313	184200	6445	136400	11345	97.011452	28
5.5	2391	219884	8324	150450	13800	96.988054	60
5.5	2379	207785	7945	155652	12639	97.073484	38
5	2157	61200	4862	19550	12990	94.456255	3
5	2154	60098	6278	18644	12678	95.340992	3
5	2138	45722	4956	18897	11980	96.982245	3
5	2125	42645	4460	17245	12208	95.345085	3
5	2120	38208	4405	14420	12055	95.967242	3

TABLE 3.9: Computational Results using Updated Simulated Annealing Method

Average patients arrival	Number of patients	Total cost	Cost of delay at Hospital	Cost of delay at home	Cost of OT	Average % between 400 day	day > 400
6	2541	668685	22060	268810	12400	97.2368311	79
6	2552	520810	26560	223950	12360	95.3056727	81
6	2659	699730	25190	363865	14950	97.1324232	105
6	2557	568225	34095	261382	12455	96.8145723	89
6	2519	359850	11875	179990	13675	97.4527551	72
5.5	2476	159655	7430	130709	10450	97.3215581	69
5.5	2300	126300	4105	104250	12700	97.0824535	25
5.5	2313	154200	5015	118960	13000	97.3215842	25
5.5	2391	209550	6875	160165	13500	97.0723713	55
5.5	2379	208680	8970	147745	12800	97.2984525	40
5	2157	59800	4485	35940	12800	95.3891437	3
5	2154	42955	6142	16400	12450	94.9339125	3
5	2138	56300	6470	30200	12940	96.7410982	3
5	2125	37350	5460	16195	11920	95.2016239	3
5	2120	35450	4300	13450	13100	95.6973218	3

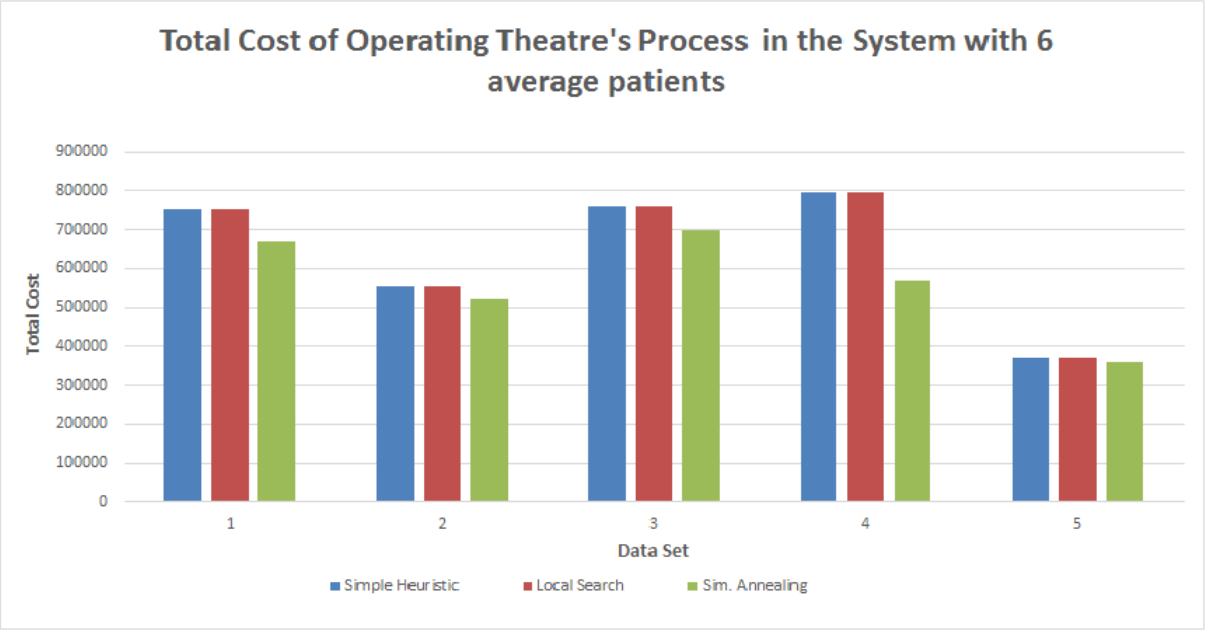

FIGURE 3.3: Total cost with 6 average patients between the methods

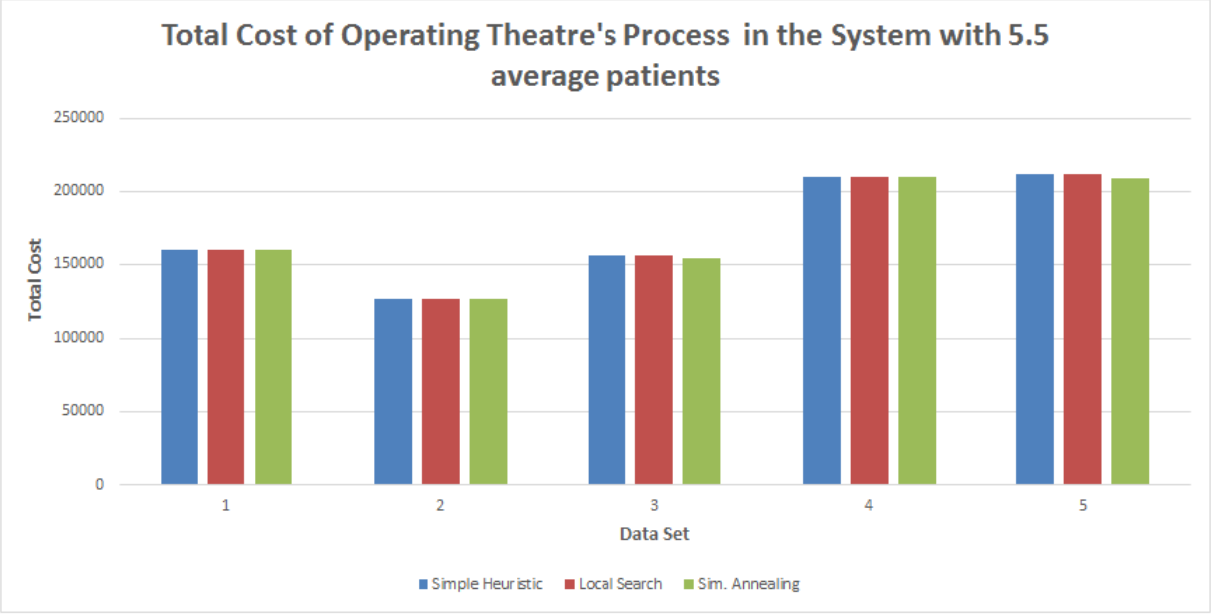

FIGURE 3.4: Total cost with 5.5 average patients between the methods

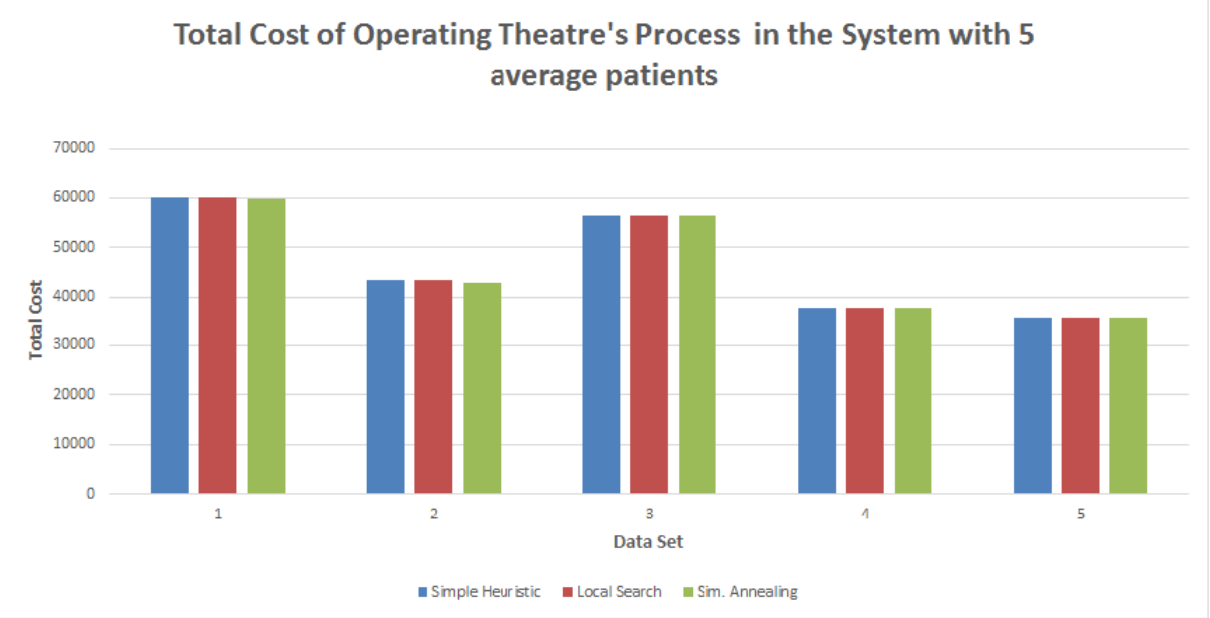

FIGURE 3.5: Total cost with 5 average patients between the methods

Figures 3.3, 3.4 and 3.5 show that there are no large differences in the total cost between the methods for each average number of patients. However, between the data sets, it can be seen that there is a different in the total cost. Figure 3.4 and 3.5 show that the total cost are almost the same between the data sets for each of the method.

Some differences can only be seen in Figure 3.6 where in four of the data sets, simulated annealing method shows some improvement compare to the other two methods. This might suggests that as the average number of patients increases, simulated annealing method might be able to reduces the total costs.

Interestingly, data set 4 in Figure 3.6 clearly shows that the total cost of OT's process in the system significantly decrease using the simulated annealing method. It also has the highest total cost for heuristics and local search method compared to the other data sets but surprisingly, the total costs for simulated annealing method is lower than the total cost of simulated annealing in data set 1 and data set 2. This might suggests that as the total costs increases for average

patients higher than 6, the simulated annealing method might be able to reduces the cost significantly.

Although Tables 3.6, 3.7, 3.9 shows that the number of total patients between the five data sets for each average number of patients to be almost the same, the same cannot be said about the the the total cost of OT's process in the system. For example, in Figure 3.3, data set 5 contains a total of 2519 patients but the total costs is only about 400000 for the three methods but on the other hand, data set 1 contains a total of 2541 patients (a difference of only 22 than data set 5) but the total cost is about 700000 for all three methods.

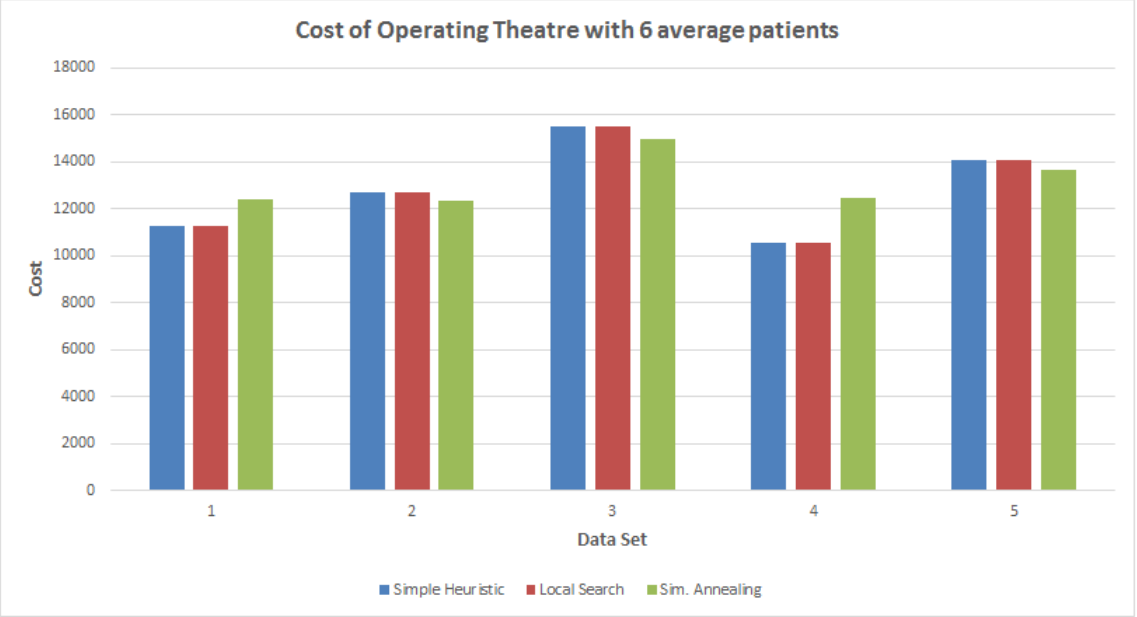

FIGURE 3.6: Cost of OT with 6 average patients

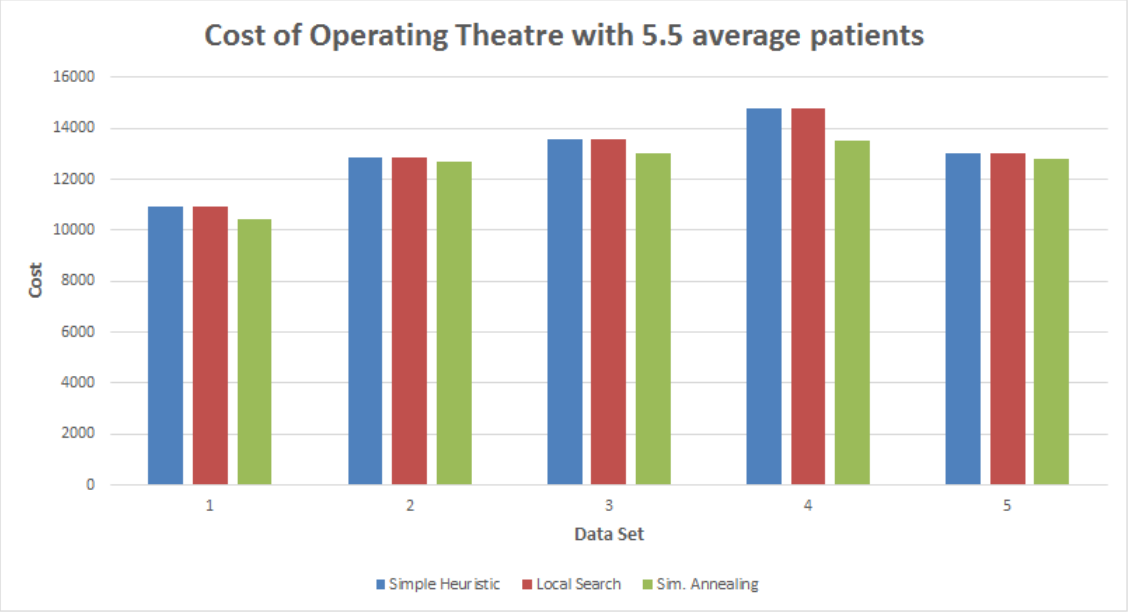

FIGURE 3.7: Cost of OT with 5.5 average patients

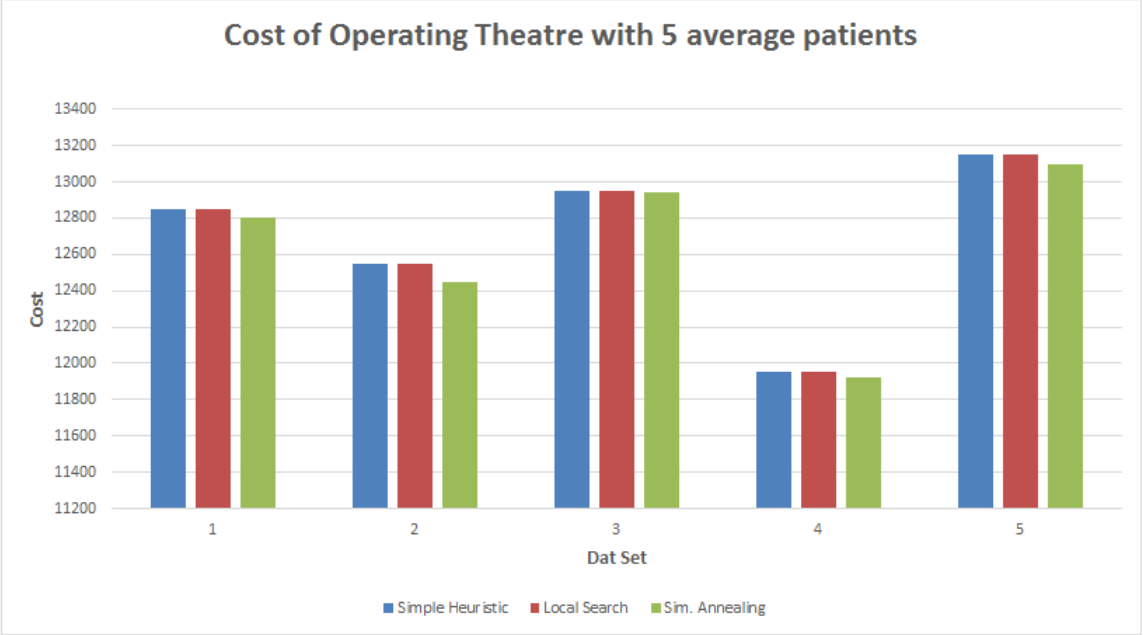

FIGURE 3.8: Cost of OT with 5 average patients

Figures 3.6, 3.7 and 3.8 show the cost of using OT over the time horizon. Again there are not much differences between the methods for each data sets but in general simulated annealing method has a little bit improvement in reducing the cost especially for 5.5 and 5 average patients. However, for 6 average patients

(Figure 3.6), we can see that the simulated annealing method actually increases the cost of OT for data sets 1 and 4 and the differences when compared to the other methods are quite substantial. This might suggest that there are many overtime use of OT under the simulated annealing method. We can also see that the cost for data set 4 in Figure 3.8 is the smallest compared to the other data sets but the difference is small.

Overall, all the data sets shows that the cost of using OT over the time horizon are more or less the same with a value of around 12000 and the highest is in data set 3 in figures 3.6 and the lowest is in data set 1 in figures 3.7. This suggests that in term of the cost of using OT over the time horizon, there is not much different. This might be because the duration time and the allocated overtime usage are similar across the data sets.

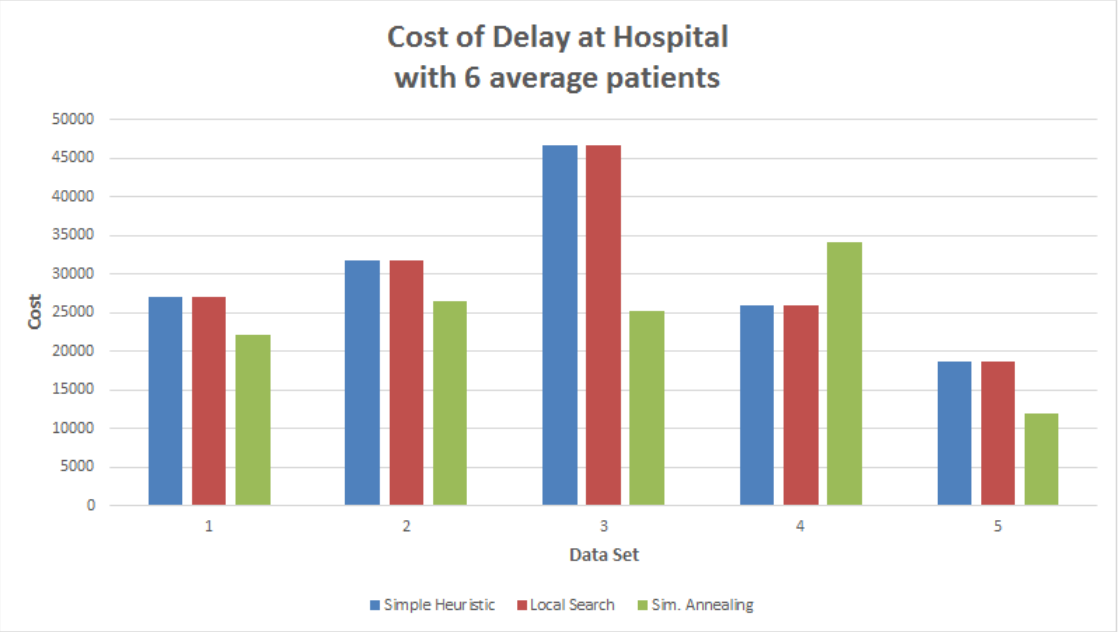

FIGURE 3.9: Cost of delay at Hospital with 6 average patients

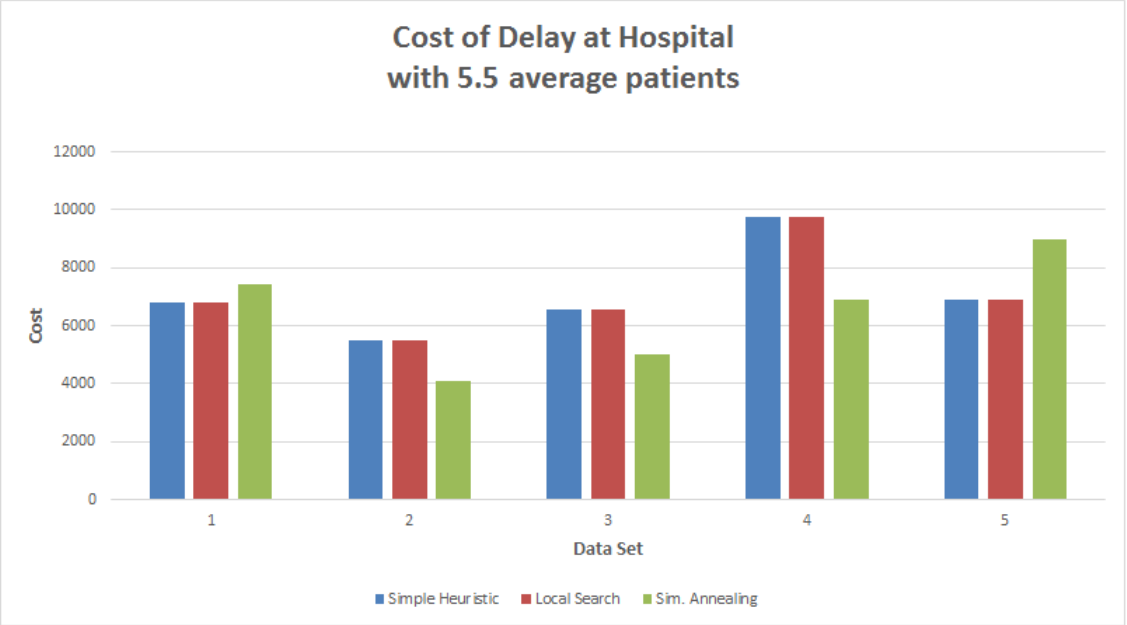

FIGURE 3.10: Cost of delay at Hospital with 5.5 average patients

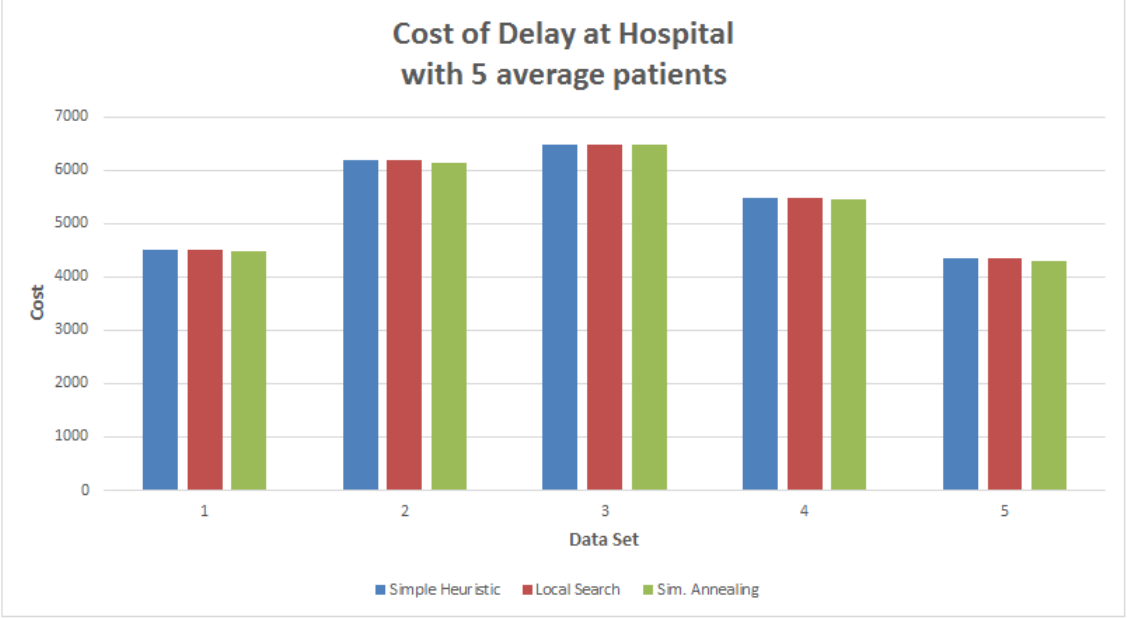

FIGURE 3.11: Cost of delay at Hospital with 5 average patients

Figures 3.9, 3.10 and 3.11 show that the cost of delay at hospital for the data sets are varied. In Figure 3.9, we can see that there are sizable difference between simulated annealing and the other two methods especially in data set 3 where the difference is about 20000. However, in data set 4, the cost under the

simulated annealing method are much bigger (an increase of about 10000 compared to the other method). Here the magnitude of the cost are big when compared to the other two figures. The lowest cost of delay at hospital is achieved by the simulated annealing method in data set 5 (the cost is 11875) and the highest is with the heuristic and local search method in data set 3 (the cost is 46645).

In Figure 3.10, the simulated annealing method incur a larger cost in two of the data sets but in the other data sets, simulated annealing method decreases the cost of delay at hospital. However, here the magnitude of the cost is small. The lowest cost of delay at hospital is achieved by the simulated annealing method in data set 2 (the cost is 4105) and the highest is with the heuristic and local search method in data set 4 (the cost is 9765).

In Figure 3.11, there are no big differences between the methods and the cost for the methods are about the same for each data set. Again the magnitude of the cost is small and the range between the cost is also small with the lowest cost is produced by data set 5 and the highest is produced by data set 3.

In can be concluded based on the cost of delay at hospital that the higher the average number of patients, the cost of delay at hospital also increases. The magnitude of the cost increases substantially even with a difference of 0.5 average patient. As we can see, when the average number of patients is 5.5, the cost is not higher that 10000 but when the average number of patients is 6, the cost goes as high up as 46000.

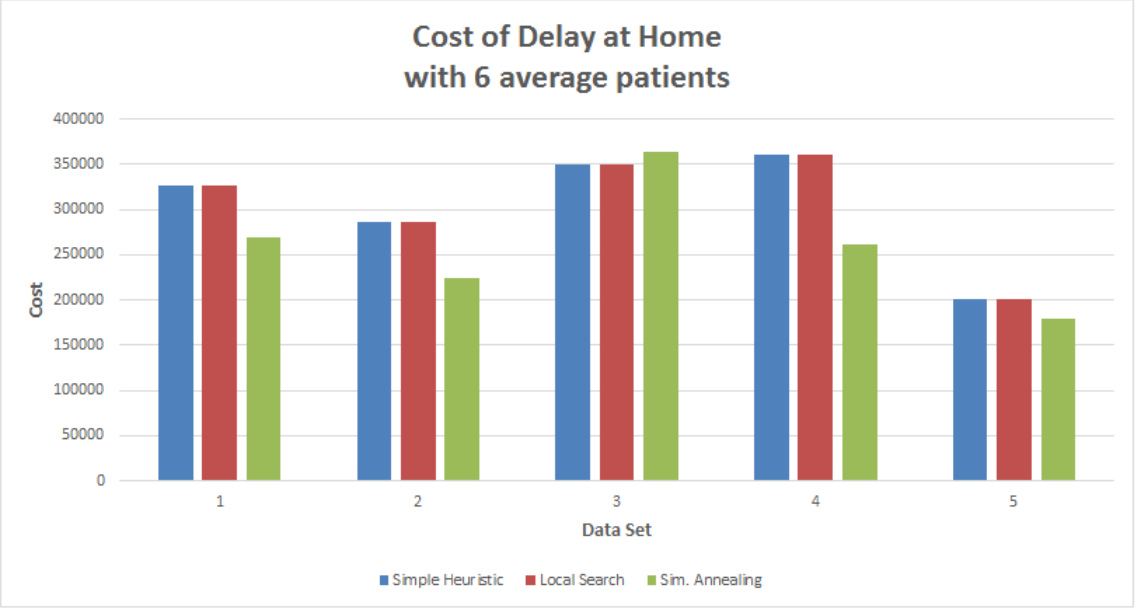

FIGURE 3.12: Cost of delay at Home with 6 average patients

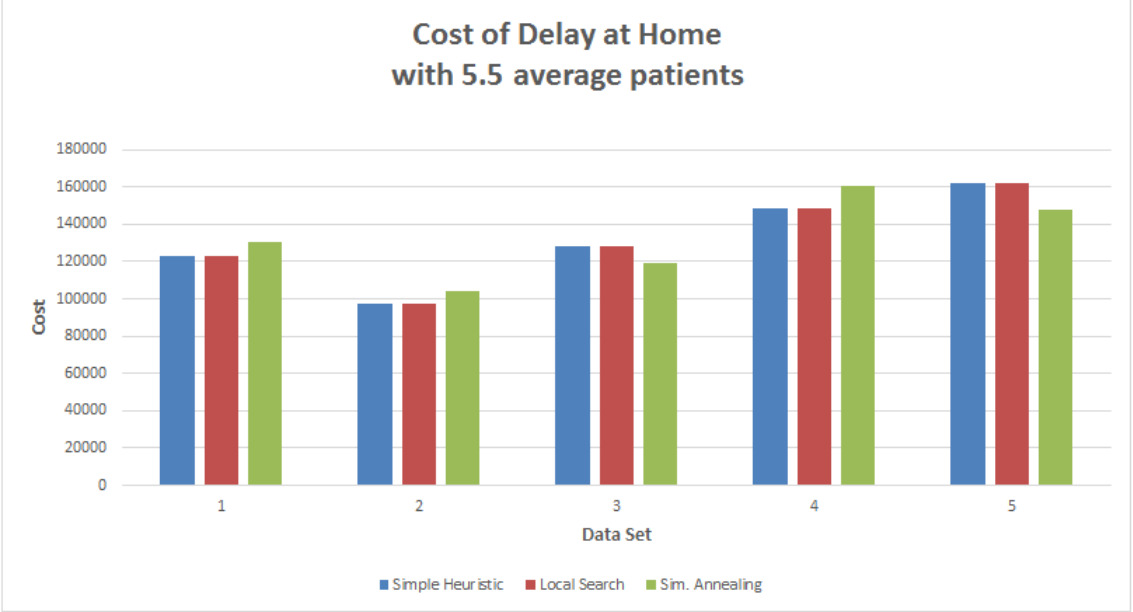

FIGURE 3.13: Cost of delay at Home with 5.5 average patients

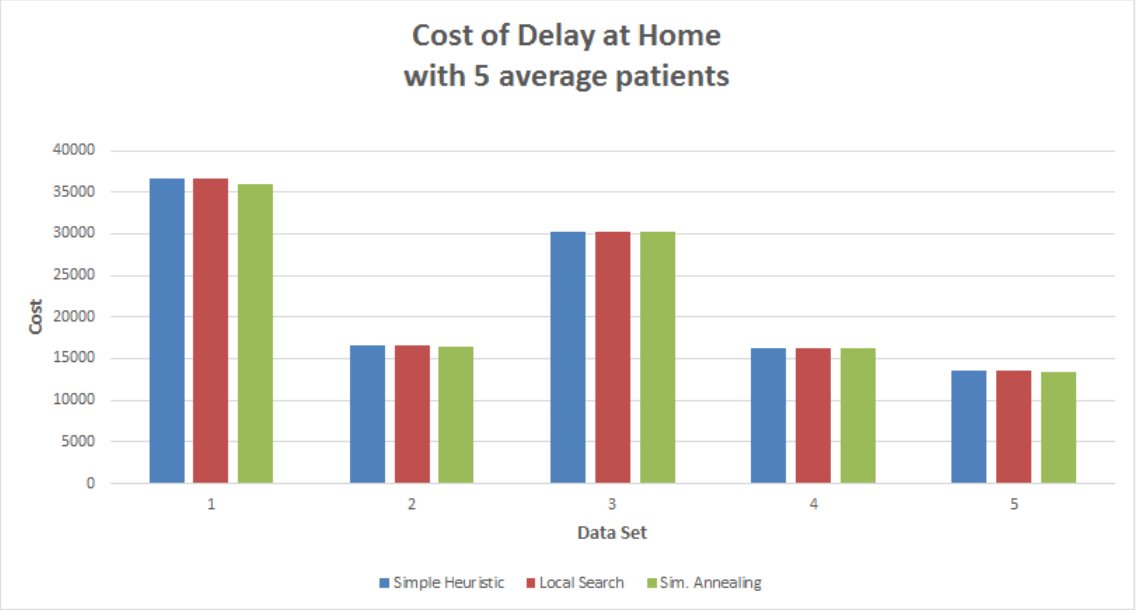

FIGURE 3.14: Cost of delay at Home with 5 average patients

Figures 3.12, 3.13 and 3.14 also show a varied data for the cost of delay at home. In some data sets, simulated annealing decreases the costs but in other data sets, it increases the cost (simulated annealing manages to reduce the cost in all data sets except data set 3). Substantial differences where simulated annealing decreases the cost can be seen in figure 3.12 data set 4 where the simulated annealing method reduces the cost by 100000 from around 350000 produced by the other two methods.

In figure 3.13, the cost of delay at home is reduced by almost half of that in figure 3.12 when the average number of patients reduces from 6 to 5.5. However, here it seems that out of the 5 data sets, simulated annealing increases the cost of delay at home in 3 of the data sets.

In figure 3.14 the cost are almost same for each data set and that the cost for the data sets seems to be in two groups where in 2 data sets the cost is at around 30000 and the other at 15000. It seems that when the average number of patients is 5, the three methods produce the same cost of delay at home. However, the

magnitude of the cost is very small suggesting that when the average number of patients is low, not many patients will be delayed at home.

Overall, it seems that the simulated annealing method is able to reduces the cost when the average number of patients is six or more but the magnitude of the cost will increase substantially. The cost doubled when the average number of patients increases from 5.5 to 6 and if the trend is true, as the average number of patients increases, the cost of delay at home will also increases. This suggests that when there are more average patients, more patients will be delayed at home.

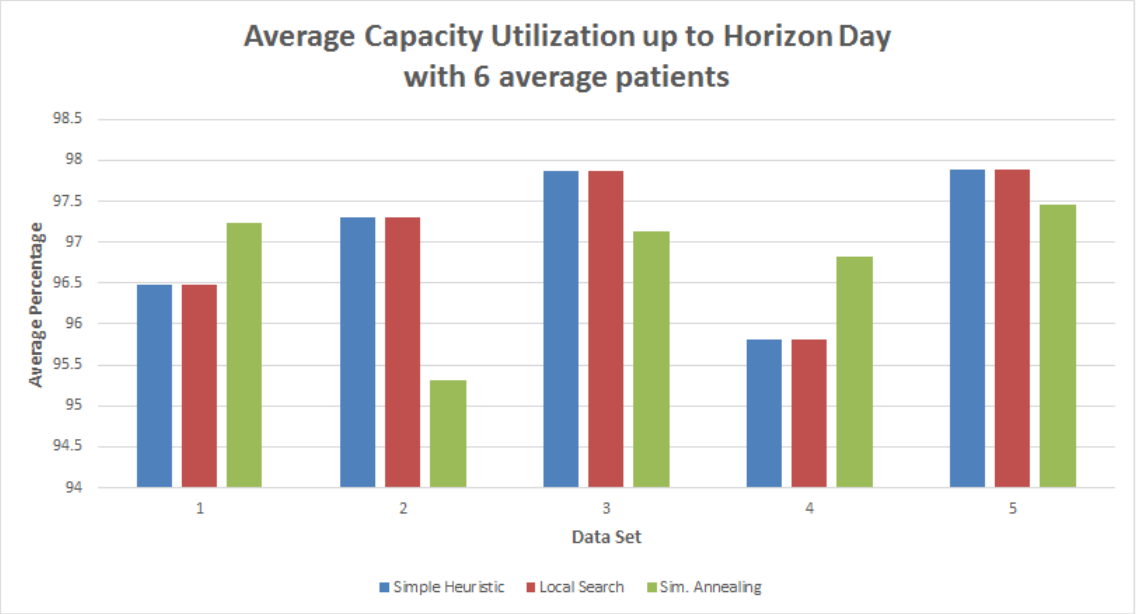

FIGURE 3.15: Average capacity Utilisation up to Horizon Day with 6 average patients

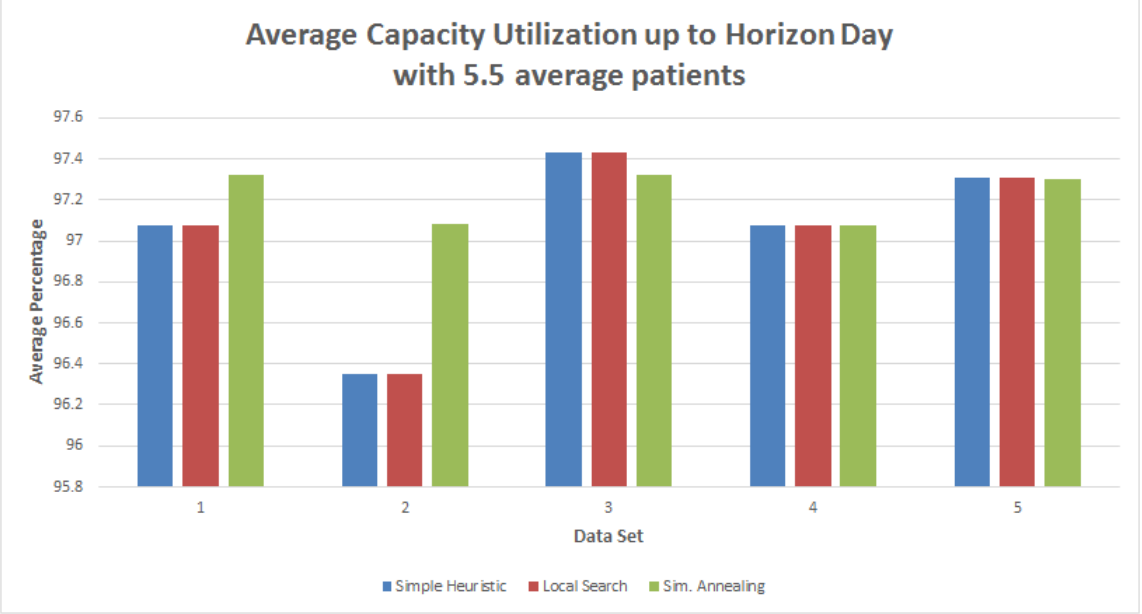

FIGURE 3.16: Average capacity Utilisation up to Horizon Day with 5.5 average patients

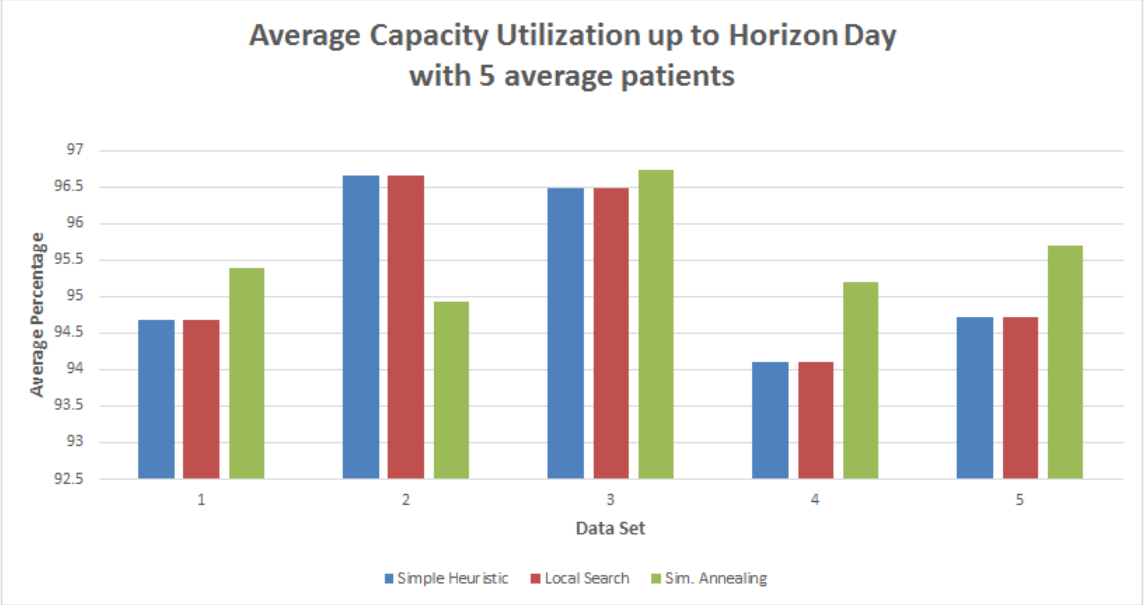

FIGURE 3.17: Average capacity Utilisation up to Horizon Day with 5 average patients

In figure 3.15, we can see that in 5 of the data sets, the average capacity utilisation is higher for the simulated annealing method in only two of them where the simulated annealing method manages to increase the utilisation by 1 percent

but reduces the utilisation in the other 3. In data set 2, the simulated annealing method reduces the utilisation by almost 2 percent.

In figure 3.16, simulated annealing method increases utilisation in only two data sets and in data set 1, the increase is only about 0.2 percent whereas in data set 2, it increases the utilisation by about 1 percent. In the other data sets, the utilisation are about the same for all three methods.

In figure 3.17, simulated annealing method manages to increase the utilisation in four data sets except for data set 2 where it reduces utilisation by about 2 percent. In the other data sets, although there is an increase but the increase is mixed with about 1 percent for data set 1, 0.5 percent for data set 3 and about 1 percent for data set 4 and 5.

Overall, there might technically be some difference between the method but since the scale is small, it could be said that the utilisation between the methods are the same. However, since the value is close to 100, it can be said that our methods are successful in utilising the OT.

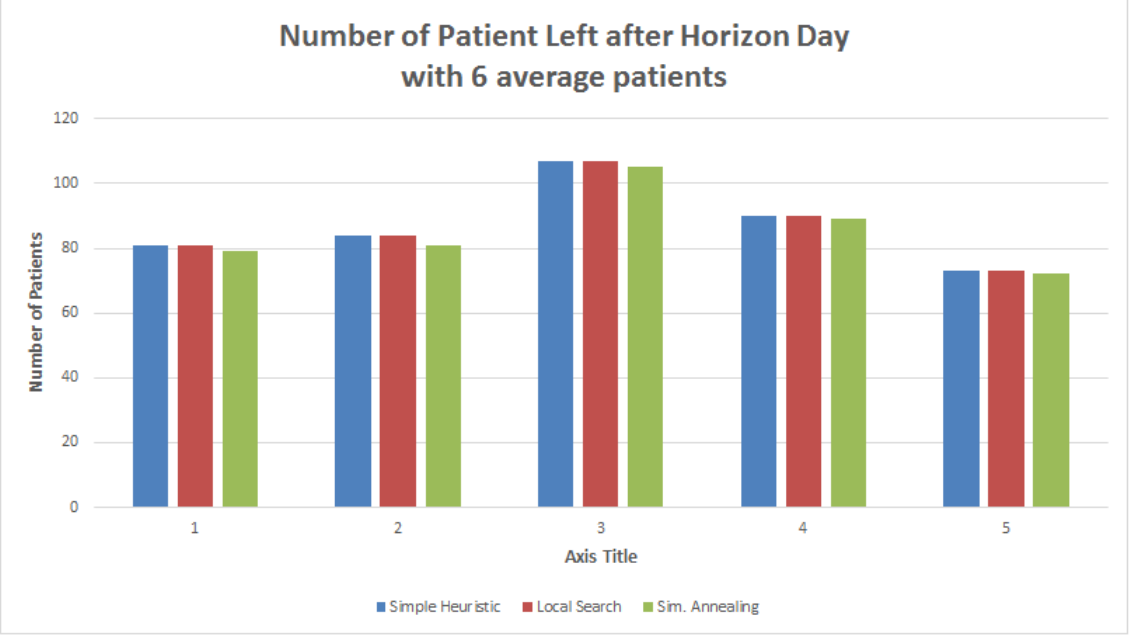

FIGURE 3.18: Number of patient Left after Horizon Day with 6 average patients

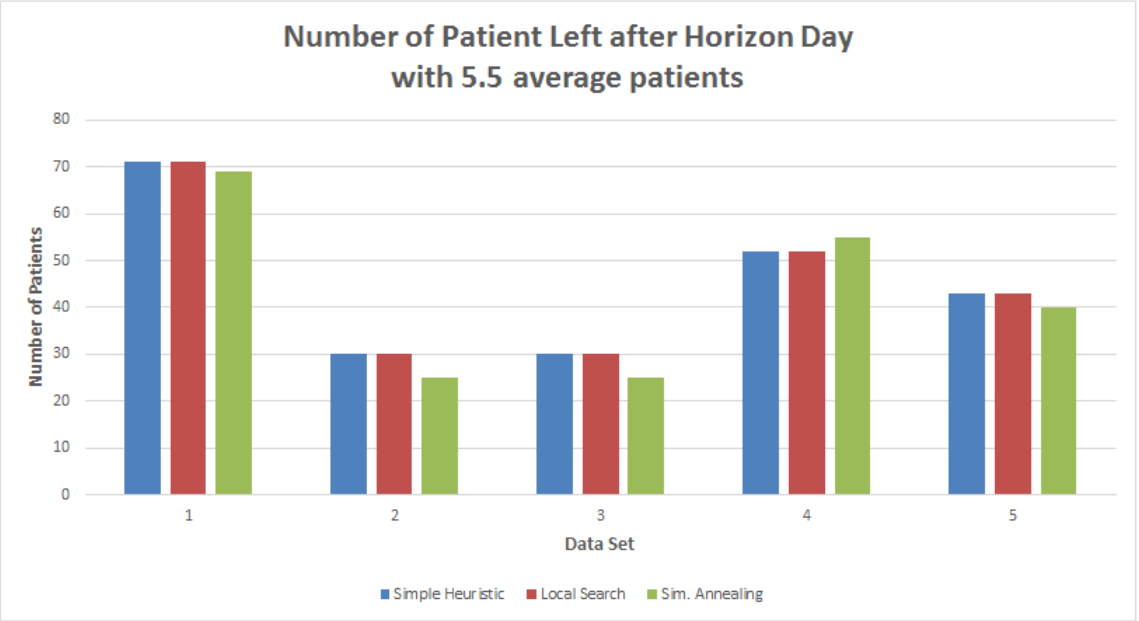

FIGURE 3.19: Number of patient Left after Horizon Day with 5.5 average patients

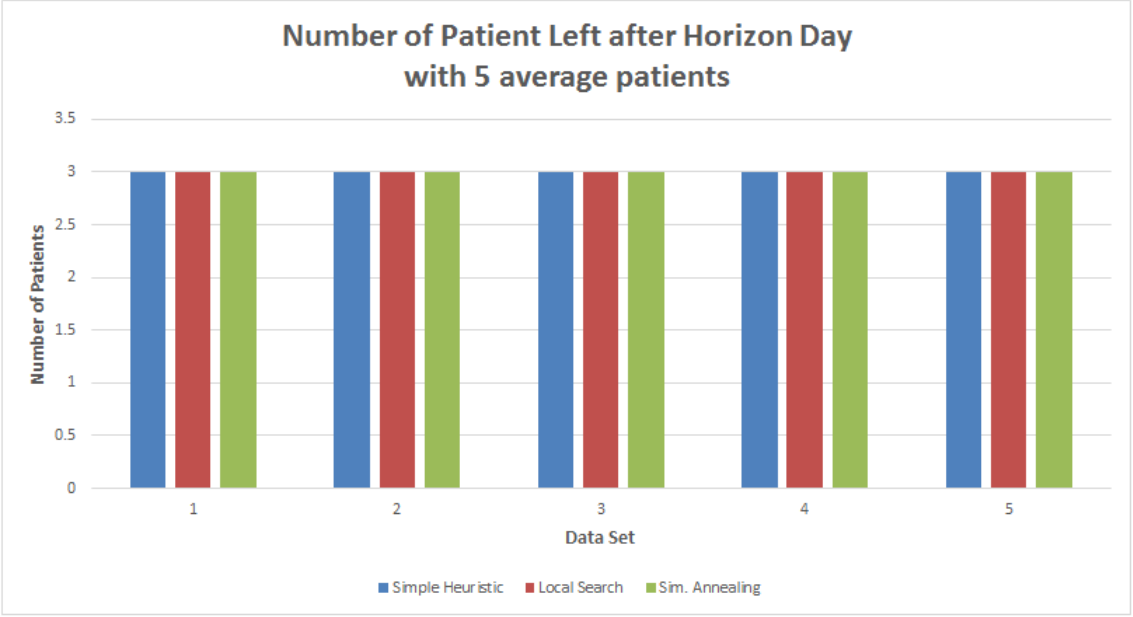

FIGURE 3.20: Number of patient Left after Horizon Day with 5 average patients

Figure 3.18 shows no differences between the methods in term of number of patients left after horizon day where on average about 80 patients left after horizon day.

Figure 3.19 shows a varied data where in data sets 2 and 3, the number of patients left is about 30 but in data set 1 the number number of patients left is around 70. However the three methods produce almost the same results.

Figure 3.20 shows no difference between the method for all data sets but the number is only 3. This suggests that when the avererage number of patients is small, the number of patients left will also be small.

3.11 Conclusion

In this chapter, we discuss the foundation of our research by defining the essential models, parameters and assumptions. We start by looking at the problem we want to tackle which is the order of scheduling of elective patients and what happens when emergency patients arrive throughout the day. We explain the delay conditions that might cause some patients to be rescheduled. We allocate all patients and reschedule delay patients depending on their case using the heuristic method in time horizon every day.

We then define the objectives of our research and the constraints considered. We also define the types of patients priority and the cost function used. The overtime cost function is presented to see how it affects our scheduling decision on whether to continue with the planned operations by doing overtime work or reschedule the patients to the next day. We also present an example of a heuristic technique and several examples of manual calculations of the cost for some patients.

The algorithms used to schedule and reschedule patients are presented. The first algorithm is the online procedure which included the algorithm for initial scheduling which will schedule the regular patients into the system. Next, we evaluate a particular day d where patients $S1$, or $S2$ or $S3$ or $S4$ are treated in

time horizon. If overtime is required, we compare the cost and penalty of being delayed at the hospital and decide whether to perform the surgery or reschedule to the next day.

Rescheduling algorithm is then applied to schedule the delayed patients with the aim of minimising the cost depending on the patients' priority. Under this algorithm, top priority patients such as patients that have been delayed several times has higher priority than those that have only been delayed once which will eliminate the possibility that urgent patients do not receive the appropriate treatment as soon as possible.

We then present the local search algorithm where we start with an initial solution from the heuristic technique, and then swap pair of patients. We swap the patients to see if by swapping, the duration and cost can be reduced. Before swapping we check sum of time on the duration day. After each swapping, the cost is calculated and compared with the current cost. If the cost is less, the swap is finalised.

Finally, we discuss the simulated annealing method where we calculate the difference of the total cost (total new cost − total current cost), Δ between a pair of patients that we plan to swap. If the difference of the total cost is positive, the swapping might still happen but only with a certain probability. Besides that, if Δ is more than zero, we swap with probability $e^{-\Delta/T}$, where T is the temperature and we have a geometric cooling $T = \alpha T$.

All the algorithms presented in this chapter will be utilised and tested with different data sets to see the best algorithm that optimised our objective in the next chapter. We wish to look for any significant difference between the methods and decide if one algorithm will be more preferred than the other.

In general, the simulating annealing method shows an improvement when compared with heuristic method and local search method. However, these values

are affected by the average number of patients and on the nature of the data sets itself. We can see the total cost is mostly made up of the cost of delay at hospital and cost of delay at home whereas the cost of OT is about the same for the different average number of patients. Also when the OT is busy (more than average patients), more rescheduling is necessary.

The utilisation percentage of the OT seems to be about the same for the different methods and it is very close to zero suggesting that there are no under utilisation of OT by the methods. Besides that, the number of patients left after time horizon is also affected by the average number of patients but this is a natural effect of the high average number of patients. When there are more patients, we would expect more delay and some patients do not get treated until after horizon day.

Lastly, different data sets produce different results because each data set contains different set of patients and are representative of the real data.

Chapter 4

On Day of Treatment Operating Theatre Scheduling Problem

4.1 Background

A different aspect of the OT scheduling problem to consider is the scheduling of multiple OTs running at the same time wherein patients can be booked into any of the OTs depending on the patient's priority and the surgeon's speciality or availability of equipments. In this chapter, we are considering the schedule on the day with the patients booked, emergency arrivals and variations from projected operation duration. The schedule on the day includes the order in which patients are treated. If the emergency arrivals or variations in operation duration exceeds the theatre slot length, the cost of either (or both) overtime or cancelling operations will be calculated. In the previous chapter, the schedule is updated at the end of the day. Whereas in this chapter, the schedule will be updated continuously every time a surgery ends and upon arrival of emergency patients.

We begin our research by reviewing the work in the area of scheduling parallel machines. The basic idea is that each OT can be considered as a machine with

jobs to be completed and under different circumstances, the methods to optimally schedule the OT will differ. Besides that, machines are available for processing jobs all the time in the planning horizon which is similar to the scheduling of OT where we assume OTs are available for surgeries in the time horizon. In addition to that, a job cannot be interrupted once it is being processed, which is a similar situation in OT occurences (surgery cannot be interrupted once it has begun).

Belouadah and Potts (1994) proposed a branch and bound algorithm in scheduling identical parallel machines to minimise total weighted completion time. They performed a Lagrangian relaxation using a noninteractive method which allows derivative of a lower bound scheme at a modest computational expense. Besides that, Hall et al. (2002) considered a deterministic scheduling of jobs on several identical parallel machines with a common server using a variety of classical scheduling objectives. They provided either a polynomial- or pseudo-polynomial-time algorithm, or a proof of binary or unary NP-completeness for each problem.

In addition, Shim and Kim (2007) considered the scheduling of parallel machine with the aim of minimising total tardiness. They also proposed a branch and bound algorithm and developed dominance properties and lower bounds as well as upper bounds from a heuristic algorithm. Computational experiments to evaluate the performance of the algorithm described in the article showed that the algorithm could find optimal solutions for problems with up to 30 jobs and 5 machines in a reasonable amount of CPU time.

Moving on from identical parallel machines, we then review some work in the area of unrelated parallel machines. For example, Liaw et al. (2003) presented a branch-and-bound algorithm and efficient lower and upper bounds are developed in scheduling independent jobs on unrelated parallel machines to minimise the total weighted tardiness. They showed that the branch-and-bound algorithm performs well on problems with up to 18 jobs and 4 machines.

Kim et al. (2003) presented search heuristics in batch scheduling of unrelated parallel machines with the objective of minimising the total weighted tardiness. They tested four search heuristics, earliest weighted due date (EWDD), shortest weighted processing time (SWPT), two-level batch scheduling heuristics (TH) and simulated annealing and found that TH and simulated annealing outperformed the other two.

In a research closer to ours, Azadeh et al. (2014) considered the scheduling of patients in emergency department laboratories having given priority to patients' treatment as determined by the triage factor to minimise the total weighted completion time. By formulating the problem as an open shop scheduling problem, they proposed a mixed integer linear programming model and developed a genetic algorithm to solve the problem. Interestingly, they applied the response surface methodology to find the optimum genetic algorithm parameters.

In the area of scheduling multiple OTs, Zhang et al. (2014) consider the dynamic assignments of a given set of surgeries to multiple identical OTs where surgeries have random durations and planned surgeon arrival times. The aim is to minimise the total expected cost incurred by surgeon waiting, OT idling and OT overtime where surgeries are assigned dynamically to OTs at surgery completion events. They proposed an efficient algorithm by combining a two-stage stochastic programming approximation and two heuristics (a one-period look ahead method and a multi-period look ahead method) to assess the cost. They showed that the dynamic scheduling significantly improves static surgery scheduling and the optimisation of the dynamic scheduling further improves the performance.

Besides that, Zhang and Xie (2015) proposed a discrete-event framework to model the surgery schedule and to evaluate the sample path gradient of a total cost incurred by surgeon waiting, OT idling and OR overtime. They used appointment scheduling for a sequence of surgeries with random duration served

by multiple identical OTs where the surgeries are assigned to OTs dynamically on a first-come, first serve basis. They showed among others that the benefits of dynamic assignment and proactive anticipation when determining appointment times are generally high and that they increase with the number of OTs and variable conditions. They emphasised the consideration of dynamic assignments in the determination of arrival time for scheduling identical surgeries that are more likely to switch OTs during the execution.

In addition, Zhao and Li (2013) considered the problem of scheduling elective surgery to multiple OTs in ambulatory surgical settings where the focus is on the daily scheduling decisions such as the number of OTs to open, the allocation of surgery-to-OT and the sequence of surgeries in each OT. The surgeries to be scheduled are known in advance, belong to different types and each OT can only perform certain surgeries. Here they assumed the setup times are sequence dependent and both setup times and surgery duration are deterministic. They proposed a Mixed Integer Nonlinear Programming (MINP) model and a Constraint Programming (CP) model with the aim of minimising the sum of fixed costs and overtime costs of the OTs. They suggested that the CP model is more efficient than the MINP model on computational time and solution quality.

4.2 Problem Definition

In this chapter, we consider the scheduling of multiple OTs in a single day only and reschedule on the day as variations from projected operation durations and emergency occur. Different amount of time due, different skills of the surgeons and different theatres might be allocated into the system. A defining approach here is to consider the random arrivals of emergency patients that take priority above all other patients. In addition to that, we consider a group of theatres being used have similar features and that all surgeries can take place in all OTs although we

can bar certain patients from entering some OTs. Only a small number of OTs are considered but with the ability to move patients to other OTs, more patients can be served.

Once emergency patients arrive, emergencies need to be fitted in as soon as possible or with priority over booked patient. This means that the patients already scheduled into the slot will be moved to either the next slot or to a different OT. If this happens, patients can be moved from one OT to another.

As mentioned before, the actual duration for each patient is likely to vary from the planning duration. Therefore, other patients may need to be rescheduled at the end of the day. The decision whether to extend the usage of OT or reschedule the last patient depends on the costs. If the cost of overtime is lower than the cost of rescheduling the patients to the next day, the overtime is utilised. Since we are scheduling for only one day, patients that were already delayed from the previous day will have a different cost associated with them in the model and this cost is calculated when considering the usage of overtime.

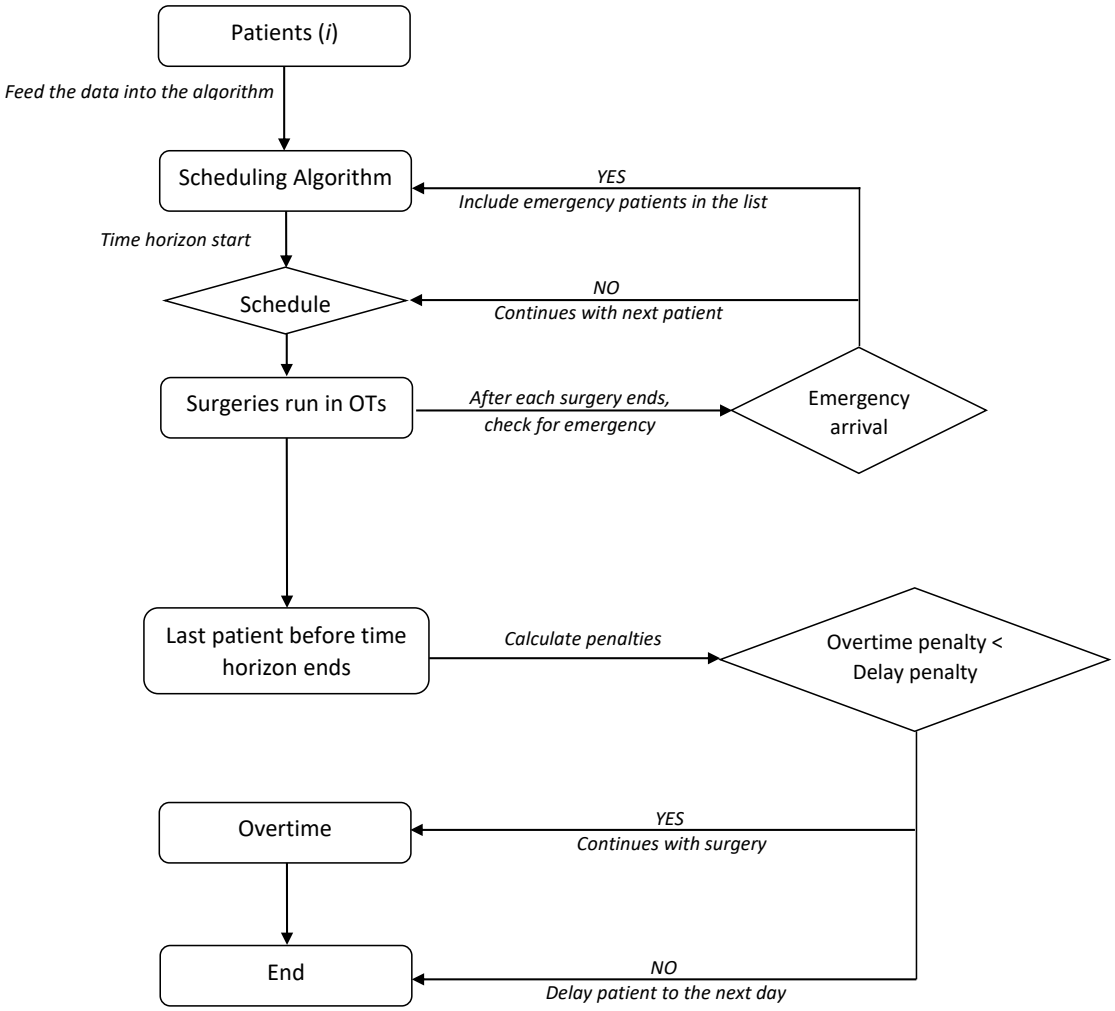

FIGURE 4.1: Initial situation

For example, as shown in Figure 4.1, at the beginning of the day we have a list of patients and they need to be scheduled to different OTs according to their respective criterion. We first feed the initial data into the scheduling algorithm and it will produce an initial schedule for the day. Then the surgeries are done based on the schedule. Since the actual duration for each surgery might be different than the planning duration, at the end of each surgery, we check if there are any emergency patients that have arrived into the system during the surgeries. If no emergency patients arrived, the surgeries go on as scheduled. But if there are

emergency patients, rescheduling is required so that the emergency patients can be operated on immediately. The data associated with the emergency patients is put into the list and then the data is fed into the scheduling algorithm to produce a new schedule. This new schedule will cause some patients to be moved from one OT to another to make way for the emergency patients.

In essence, we actually consider if rescheduling is required after the completion of every surgery provided that emergency patients arrive within the time horizon. If there are no emergency patients, then the schedule will continue in the original order. Beside that, due to the variations in surgery durations, rescheduling is also required to ensure that patient that can fit into the available slots of other OTs, if any, are scheduled to the new OTs to prevent overtime or delay.

At the end of the time horizon, rescheduling might cause overtime usage of the OT. We need to decide whether to perform the surgery with overtime usage of the OT or cancel the patients for the day and rebook them for another day. The decision is based on the comparison of the overtime cost and rescheduling cost. If the overtime cost is lower than rescheduling cost, overtime is utilised.

4.3 The Model

The surgery of n patients must be scheduled in m OTs, during time horizon for one day. Each OT has different surgeons. There are also variations in the type of equipments available in each OT. This means that some surgeries can only be performed in some OTs.

We assume that the patients who are scheduled for a particular day have already been hospitalized. We also consider priorities of patients based on their weight by giving top priority patients a higher weight. We also have disruptions from a small number of emergency patients, as some patients may be moved from

one OT to another OT or surgeries are taking longer than the planning duration. We reschedule as necessary when emergencies arrive into the system or operation durations are different from predicted. Our objective is to minimise the cost of the new schedule. The costs are penalty for patients not treated on the day and penalty for schedule overrunning at OTs. Decision variable is x_{ij} and binary variables are υ_j, z_{i1} and z_{i2}.

The notations used that relate to the environment in this study are as follows:

i: Operation theatre

j: Patient

n: The number of patients (including emergency patients once they have arrived)

m: The number of OT

T: time horizon

T_1: The upper bound of theatre time for OT i after T time horizon

T_2: The upper bound of theatre time for OT i after T_1 time horizon

4.3.1 Objective function

Our objective is to minimise the costs of the new schedule.

The costs are given as follows:

1. Penalty for patient not treated on the day.

2. Penalty for schedule overrunning at OTs (small penalty and large penalty).

4.3.2 Decision and Binary Variables

The decision variables x_{ij} are the variables that show whether patient j is assigned to one the OTs. It will be *1* if it is assigned to a particular OT and *0* in other

OTs to signal that the patients have been assigned an OT therefore it cannot be considered for the other OTs. The binary variables υ_j are the variable that show if patient j is untreated or has been treated on the day being considered. It will be *1* if the patient is untreated and *0* if it has been treated. The binary variables z_{i1} and z_{i2} show the usage of overtime where z_{i1} will be *1* if overtime band 1 is used and if overtime band 2 is used, z_{i2} will be *1*. If overtime band 2 is used, the variable z_{i1} will be *0* since overtime band 1 is included into z_{i2}. Logically, overtime band 2 cannot be used unless overtime band 1 has already been used.

The binary variables are z_{i1} and z_{i2}.

$$x_{ij} = \begin{cases} 1 & \text{if patient } j \text{ is assigned to operation theatre } i \\ 0 & \text{otherwise.} \end{cases}$$

$$\upsilon_j = \begin{cases} 1 & \text{if patient } j \text{ untreated on the day being considered} \\ 0 & \text{otherwise.} \end{cases}$$

$$z_{i1} = \begin{cases} 1 & \text{if theatre } i \text{ completes its schedule in overtime band 1} \\ 0 & \text{otherwise.} \end{cases}$$

$$z_{i2} = \begin{cases} 1 & \text{if theatre } i \text{ completes its schedule in overtime band 2} \\ 0 & \text{otherwise.} \end{cases}$$

4.3.3 Constraints

We consider several constraints in our model, the constraints are given as follows:

1. Each surgery is either assigned to exactly one OT or will be untreated for the day.

2. The overtime of OT, i.

3. Include at most one of the overtime bands.

4.3.4 Zero-One Programming Model

The notations used that relate to the variables in the integer programming model are as follows:

w_j: The weight of penalty patient j if patient j is not treated the day being considered

β_{i1}: The penalty of completion time for OT j after time horizon in time section 1, $(T, T_1]$

β_{i2}: The penalty of completion time for OT j after time horizon in time section 2, $(T_1, T_2]$

x_{ij}: Decision variable for patient j if assigned to operation theatre i

υ_j: Binary variable for patient j if untreated on the day being considered

z_{i1}: Overtime band 1 of theatre i

z_{i2}: Overtime band 2 of theatre i

$$min(\mathbf{P}) = \sum_{j=1}^{n} w_j \upsilon_j + \sum_{i=1}^{m} (\beta_{i1} z_{i1} + \beta_{i2} z_{i2})$$

subject to

$$\sum_{i=1}^{m} x_{ij} + \upsilon_j = 1, \qquad \forall j = 1, 2, \ldots, n \tag{4.1}$$

$$\sum_{j=1}^{n} t_{ij} x_{ij} \leq T + (T_1 - T) z_{i1} + (T_2 - T) z_{i2}, \qquad \forall i = 1, 2, \ldots, m \tag{4.2}$$

$$z_{i1} + z_{i2} \leq 1, \qquad \forall i = 1, 2, \ldots, m \tag{4.3}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i = 1, 2, \ldots, m, \qquad \forall j = 1, 2, \ldots, n \tag{4.4}$$

$$z_{i1}, z_{i2} \in \{0, 1\}, \qquad \forall i = 1, 2, \ldots, m \tag{4.5}$$

$$\upsilon_j \in \{0, 1\}, \qquad \forall j = 1, 2, \ldots, n \tag{4.6}$$

The objective is to minimise P where it has two components. The first component is the cost if the patients are untreated on the day considered, $\sum_{j=1}^{n} w_j \upsilon_j$. It is the product of patient j weight w_j and the binary variable υ_j. As stated earlier, the binary variable υ_j will be *1* if the patient is untreated and hence it will be multiple by the weight and counted for every patient untreated. The more patients untreated the larger the sum will be especially if the weight is big. The second component is the cost of overtime usage, $\sum_{i=1}^{m}(\beta_{i1} z_{i1} + \beta_{i2} z_{i2})$. As stated earlier, if overtime band 1 is used, the variable z_{i1} will be *1* and it will be multiplied by the overtime cost β_{i1}. If overtime band 2 is used, the variables z_{i1} will be *0* and z_{i2} will be *1* and it will be multiplied by the overtime cost β_{i2} since the cost of overtime band 1 is already included into the overtime cost β_{i2}.

The constraint $\sum_{i=1}^{m} x_{ij} + \upsilon_j = 1$ ensures that the patient j is either assigned for surgery in one of the OTs or is delayed into the next. The constraint $\sum_{j=1}^{n} t_{ij} x_{ij} \leq T + (T_1 - T) z_{i1} + (T_2 - T) z_{i2}$ ensures that the sum of the planning durations for each patient in each OT is less than or equal to the available time horizon T or if overtime is used, less than the overtime period and not more than that. The constraint $z_{i1} + z_{i2} \leq 1$ ensures that only one overtime band is activated to prevent double counting of costs since the costs for overtime band 1 is already included in overtime band 2. $x_{ij} \in \{0, 1\}$, $z_{i1}, z_{i2} \in \{0, 1\}$ and $\upsilon_j \in \{0, 1\}$ ensure the variables will always take the value of either *1* or *0* as explained in the previous section.

4.3.4.1 Overtime Costs

The penalty for time horizon greater than T:

$$\beta = \begin{cases} \beta_{i1} & T < C_i \leq T_1 \\ \beta_{i2} & T_1 < C_i \leq T_2 \end{cases}$$

We schedule the number of patients, N at time horizon, T minutes in M OTs. Some patients maybe moved from one OT to another OT because we reschedule again when we have disruption from emergency patients coming into the system or some patients have different (longer) actual durations from the predicted duration time.

Some patients can also be cancelled from the list on the day because high-priority emergency patients are scheduled into the slots. We fix T minutes in time horizon every day. If the sum of the duration time is greater than T minutes, the overtime usage of the OT will get a penalty depending on the slot time after time horizon. Also, if a patient is cancelled from the scheduling list of the day, we get a penalty.

4.4 Algorithm Design

The algorithm is designed to input and update all data before it is fed into the Zero-One Programming (ZOP) model. Every time a trigger occurs (the end of surgery or the arrival of emergency patient), the data is updated to reflect the current situation in the OTs. Some surgeries could finish early which will free up space for suitable patients or it could take longer than the planning duration which might cause delay. The arrival of emergency patients that require instant surgeries will trigger the algorithm to feed the current updated data into the ZOP model, thus producing a new schedule that will include reassignment of patients to a new OT or the delay of patients.

The notations used in the algorithm are as follows:

L: Set of patients

L^e: Set of arrived emergency patients

L_j: Set of patients assigned to operation theatre, j

M: Set of operation theatres

$\pi_j(i)$: The sequence of patients' for theatre, j

n_j: The number of patients assigned to operation theatre, j

A: List of sum of time for each operation theatre for one day

t_{ij}: The actual surgical duration time of patient, i at operation theatre, j

t_{ij}^p: The planning surgical duration time of patient, i at operation theatre, j

r_i: Release time for patient, i (this will be 0 for non-emergencies and the arrival time for emergencies)

T^c: Change time

T^e: Arrival time for emergency patient

T^r: Removal time of patient from the list (Surgery End time)

C_i: Completion time of OT i is completed

β_{j1}: The penalty of completion time for OT j after time horizon in time section 1, $T < A \leq T_1$

β_{j2}: The penalty of completion time for OT j after time horizon in time section 2, $T_1 < A \leq T_2$

Algorithm 4.1 Parallel Scheduling Algorithm

1: Input the set of operation theatres $M = \{1, ..., m\}$ and the set of patients $L = \{1, ..., n\}$, together with r_i, t^p_{ji} for $j = 1, ..., m$ and w_i, for $i = 1, ..., n$.
Set $\overline{L} = L, T^r = 0, \overline{v}_i = 1$ and $\overline{w}_i = w_i$ for $i \in L$, and $\overline{z}_{j1} = \overline{z}_{j2} = 0$.

2: If $L \neq \emptyset$, solve the IP model Zero-One Programming Model

3: If $L = \emptyset$, or if $L \neq \emptyset$ and $x_{ji} = 0$ for all $j \in L$ and $i \in M$, then apply the following:
Wait for the first arrival in the interval $(T^r, T + T^r]$ and if there are none, then go to 7.
If the first arrival is for patient i, then input r_i, t^p_{ji} for $j = 1, ..., m$ and w_i, and set $\overline{L} = \overline{L} \cup \{i\}, \overline{v}_i = 1, \overline{w}_i = w_i, T^c = r_i - T^r, L = L \cup \{i\}, T = T - T^c, T_1 = T_1 - T^c, T_2 = T_2 - T^c$ and $T^r = r_i$, and go to 2.

4: Apply the following for $j = 1, ..., m$.
Set $n_j = \sum_{i \in L} x_{ji}$ and $L_j = \{i \in L | x_{ji} = 1\}$.
If $L_j \neq 0$, form a sequence $\pi_j = (\pi_j(1), \pi_j(2), ..., \pi_j(n_j))$ for the patients in L_j such that $w_{\pi_j(1)} \geq w_{\pi_j(2)} \geq ...w_{\pi_j(n_j)}$.
If $L_j \neq \emptyset$, set $w_{\pi_j(1)} = \infty, t^p_{j',\pi_j(1)} = \infty$ for $j' \in M \backslash \{j\}$, and $\overline{v}_{\pi_j(1)} = 0$.

5: Wait for the first patient among $\{\pi_j(1) | j \in M, L_j \neq \emptyset\}$ to complete surgery or the next emergency patient to arrive, whichever event occurs first.
If the first event is the arrival of a new patient i, then input r_i, t^p_{ji} for $j = 1, ..., m$ and w_i, and set $\overline{L} = \overline{L} \cup \{i\}, \overline{v}_i = 1, \overline{w}_i = w_i, T^c = r_i - T^r$, and $T^r = r_i$.
If the first event is the completion of the operation for patient $\pi_j(1)$ at time $C_{\pi_j(1)}$, then set $T^c = C_{\pi_j(1)} - T^r$ and $T^r = C_{\pi_j(1)}$.
Set $T = T - T^c, T_1 = T_1 - T^c, T_2 = T_2 - T^c$.

6: For each $j \in M$ with $L_j \neq \emptyset$ and $C_{\pi_j(1)} = T^r$, set $L_j = L_j \backslash \{\pi_j(1)\}$; if $T + T^r < C_{\pi_j(1)} \leq T_1 + T^r$, then set $\overline{z}_{j1} = 1$; and if $C_{\pi_j(1)} > T_1 + T^r$, then set $\overline{z}_{j1} = 0$ and $\overline{z}_{j2} = 1$.
For each $j \in M$ with $L_j \neq \emptyset$ and $C_{\pi_j(1)} > T^r$, set $t^p_{j,\pi_j(1)} = max\{t^p_{j,\pi_j(1)} - T^c, 0\}$.
Go to 2

7: Compute the total cost $\sum_{i \in \overline{L}} \overline{w}_i \overline{v}_i + \sum_{j \in M} \beta_{j1} \overline{z}_{j1} + \beta_{j2} \overline{z}_{j2}$

In step 1, we must set the number of OTs available for surgeries. We then set the number of patients available for surgeries with their arrival time and weight. For each patient, their planning duration in each of the OTs available is known and set. Once all the relevant data is available, a list of available patients for

the day is now known. For the initial set up, we must set the removal time of patient from the list (surgery End time) $T^r = 0$ since no surgery has taken place. In step 2, with the list containing all available data, we feed it into the Zero-One Programming (ZOP) model and it will choose the OT for each patient for the day.

In step 3, if the list of patients is not empty but one or more OTs is empty and available for surgery in the time horizon we will wait for the arrival of new patient (emergency patient) in the interval between last patient removed and end of time horizon. If a new patient arrives, we input the data for patient and update the list before feeding it into the IP model. If no patient arrives, then the OT is closed at the end of the time horizon.

In step 4, we will know the number of patients assigned to each OT. In each OT, the list of patients is sorted according to their weight with highest weight scheduled for surgery first. Once the surgery has started, the weight for the patient and the planning duration in all other OTs are set to an arbitrary large value to prevent interruption. Once a surgery has started it must continue until it is completed. Once it has finished, the patient is removed from the list.

In step 5, we wait for the first surgeries among the OTs to finish or the next emergency patient to arrive, whichever occurs first. If a new patient arrives during the surgeries, it will trigger an update to the list to include the data for the new patient and the updated list is fed into the IP model to produce a new schedule. This new schedule will prioritise the new emergency patient above all other so the patients will be booked for surgery after one the current surgeries has finished to be bumped into the slot or to other OTs. On the other hand, if a surgery finishes first, then the data in the list is also updated.

In step 6, after each trigger, patients that have finished surgery is removed from the list and the time horizon is updated to reflect the available time left for surgery. If the time horizon has been reached then overtime band is taken into

consideration for usage. This will inform the IP model whether to schedule or reschedule patients to the next day by calculating their costs.

Finally, in step 7, if there are no more surgeries scheduled, we calculate the total cost.

4.5 Example of Linear Programming Solving

We generate example data sets for one day with 2 OTs and 12 total number of patients, where 10 regular patients and 2 emergency patients arriving during the day. We focus on the order in which patients are treated for OT Scheduling Problem. The main challenge ocurs when emergency patients arrive online in the system. Every day, we schedule patients in the empty slots of OT time horizon based on the current information available. The schedule is updated every time a surgery is completed and when the emergency patients arrive into the system. A high number of emergency patients arriving into the system and operation times longer than expected can make other patients move to another OT or they will be cancelled on that day.

Table 4.1 shows the list of patients, L that are scheduled for the day (patients 1 to 10) and the emergency patients, L^e that will arrive during the day. Each patient has with them their planning duration and actual duration in each OT and the weight w_j. Table 4.2 shows the time horizon where if overtime is required (T_1 and T_2), we set two horizon for overtime and the value of penalty of overtime usage in this system.

With this list of patients, we feed the data into the IP model and it will sort the patients into the OTs such that the patients are scheduled into the best

TABLE 4.1: Example Data for 12 patients on one day.

Patient	**Planning Duration**		**Weight**	**Actual Duration**		**Emergency time arrival**
1	80	1000	10	100	1000	10
2	130	800	20	140	800	0
3	60	560	15	70	560	0
4	60	990	30	90	90	0
5	110	95	20	120	80	0
6	500	100	15	500	130	0
7	1000	90	40	1000	90	0
8	710	110	25	710	100	0
9	100	80	15	90	100	0
10	90	1000	40	130	1000	0
11	50	50	60	60	50	100
12	70	200	60	80	200	200

TABLE 4.2: Example Data for Services of Operation Theatre.

Time Horizon T	480 minutes
Upper bound 1 T_1	550 minutes
Upper bound 2 T_2	700 minutes
Cost Penalty 1	50
Cost Penalty 2	100

OTs that minimise costs and maximise the number of patients scheduled. The output from the IP model is shown in Table 4.3. We can see that five patients are scheduled in OT 1 and five patients are scheduled in OT 2 where the total planning duration for OT 1 is 420 minutes and the total planning duration for OT 2 is 475.

Patients are sorted according to their respective weight to ensure that patients with higher weight will be operated first, since the weight represent the urgency of the surgery. Once this is done, we will then have the schedule for the day and the surgeries can start as shown in Table 4.4.

Once the surgeries are running in the OTs, we wait to see if any emergency patient arrives, which will require us to reset the schedule. Also, we wait to see if any surgery ends, which indicates that actual duration differs from planning

TABLE 4.3: List of Initial Schedule.

Patient	**Operation Theatre**	**Planning Duration**		**Weight**	**Actual Duration**	
1	1	80	1000	10	100	1000
2	1	130	800	20	140	800
3	1	60	560	15	70	560
4	1	60	990	30	90	990
10	1	90	1000	40	130	1000
Total OT 1		420				
5	2	110	95	20	120	80
6	2	500	100	15	500	130
7	2	1000	90	40	1000	90
8	2	710	110	25	710	100
9	2	100	80	15	90	100
Total OT 2			475			

TABLE 4.4: Sorted based on weight.

Patient	**OT**	r_i	**Planning Duration**		**Weight**	**Actual Duration**	
10	1	0	90	1000	40	130	1000
4	1	0	60	990	30	90	990
2	1	0	130	800	20	140	800
3	1	0	60	560	15	70	560
1	1	0	80	1000	10	100	1000
Total			420				
7	2	0	1000	90	40	1000	90
8	2	0	710	110	25	710	100
5	2	0	110	95	20	120	80
6	2	0	500	100	15	500	130
9	2	0	100	80	15	90	100
Total			475				

duration. Some surgeries might be shorter than the planning duration or it might take longer due to complication during the surgery. In our example in Table 4.5, the first patient finishes after 90 minutes, which is patient 7 in OT 2. Following this, Patient 7 is no longer considered for scheduling. However, Patient 10 is still in surgery and continues in the schedule.

Hence at $T^c = 90$, we adjust the remaining time available $T = 390$, $T_1 = 460$

and $T_2 = 610$. From Table 4.6, we then continue with patient 8 in OT 2 since the order of scheduling has not changed. However, for Patient 10 in OT 1, we greatly increases the weight because as it has already started, it must be completed and cannot be interrupted even with the arrival of an emergency patient. Hence the surgery for Patient 8 starts.

TABLE 4.5: Operation Slot 1

Patient	**OT**	r_i	**Planning Duration**		w_i	**Slot 1**			**Time start**
10	1	0	90	1000	40	130	1000	continue	
4	1	0	60	990	30	90	990		
2	1	0	130	800	20	140	800		
3	1	0	60	560	15	70	560		
1	1	0	80	1000	10	100	1000		
Total			420						
7	2	0	1000	90	40	1000	90	Finish	90min
8	2	0	710	110	25	710	100		
5	2	0	110	95	20	120	80		
6	2	0	500	100	15	500	130		
9	2	0	100	80	15	90	100		
Total			475						

TABLE 4.6: Next Operation Start

Patient	**OT**	r_i	**Planning Duration**		w_i	**Actual Duration**		**Slot 1**	**Time start**
10	1	0	40	2000	500	130	1000	continue	90min
4	1	0	60	990	30	90	990		
2	1	0	130	800	20	140	800		
3	1	0	60	560	15	70	560		
1	1	0	80	1000	10	100	1000		
Total			370						
7	**2**	**0**	**1000**	**90**	**40**	**1000**	**90**	**Finish**	
8	2	0	710	110	25	710	100	Start	90min
5	2	0	110	95	20	120	80		
6	2	0	500	100	15	500	130		
9	2	0	100	80	15	90	100		
Total				385					

After $T^c = 90$, the surgeries continue and similarly we wait for the arrival of emergency patients. The example in Table 4.6 shows emergency patient 11 arrives into the system at $T^e = 100$ minutes. The data associated with Patient 11 is added into the system and we update $T^c = 100$, $T = 380$, $T_1 = 450$ and $T_2 = 600$ since the arrival of emergency will prompt us to input the new updated data into our IP model and it will produce a new schedule as shown in Table 4.7.

TABLE 4.7: New Schedule after patient 11 arrived

Patient	**OT**	r_i	**Planning Duration**		w_i	**Actual Duration**		**Slot 1**	**Time start**
10	1	0	30	1000	500	130	1000	Continue	100min
11	1	10	50	50	60	60	50	New	
4	1	0	60	990	30	90	990		
2	1	0	130	800	20	140	800		
3	1	0	60	560	15	70	560		
1	1	0	80	1000	10	100	1000	Delay	Delay
Total			330						
7	**2**	**0**	**1000**	**90**	**40**	**1000**	**90**	**Finish**	
8	2	0	710	100	500	710	100	Continue	100min
5	2	0	110	95	20	120	80		
6	2	0	500	100	15	500	130		
9	2	0	100	80	15	90	100		
Total				375					

In Table 4.7 we can see that Patient 11 is now scheduled right after patient 10 has finished surgery. In addition to this, the weight for patient 8 is updated to 500 since the surgery is now in progress and cannot be interrupted. With the addition of Patient 11, the IP model decides that Patient 1 will be delayed since the planning duration now exceeds 380 if Patient 1 is included and the cost of delay is lower than the cost of overtime.

After the new schedule is produced, the surgeries go on as planned and without the arrival of emergency patients, the schedule is not updated and soon arrives at 130 minutes where now the surgery for patient 10 is finished and hence the schedule is updated where $T^c = 130$, $T = 350$, $T_1 = 420$ and $T_2 = 570$. Then

the next surgery follows, where Patient 11 is operated in OT 1 and patient 8 continues without interruption as shown in Table 4.8

TABLE 4.8: Schedule at 130 minutes

Patient	**OT**	r_i	**Planning Duration**		w_i	**Actual Duration**		**Slot 1**	**Time start**
10	**1**	**0**	**30**	**1000**	**500**	**130**	**1000**	**Finish**	
11	1	10	50	50	60	60	50	Start	130min
4	1	0	60	990	30	90	990		
2	1	0	130	800	20	140	800		
3	1	0	60	560	15	70	560		
1	1	0	80	1000	10	100	1000	Delay	Delay
Total			300						
7	**2**	**0**	**1000**	**90**	**40**	**1000**	**90**	**Finish**	
8	2	0	710	70	500	710	100	Continue	130min
5	2	0	110	95	20	120	80		
6	2	0	500	100	15	500	130		
9	2	0	100	80	15	90	100		
Total				345					

The surgeries continue without interruption until 190 minutes where surgeries for Patient 11 and Patient 8 are finished. Hence the schedule is updated where $T^c = 190$, $T = 290$, $T_1 = 360$ and $T_2 = 510$. The next surgeries go on as scheduled for Patient 4 in OT 1 and Patient 5 in OT 2 with the remaining planning duration of 250 for OT 1 and 275 for OT 2. Next, at 200 minutes, Patient 12 arrives into the system and the list is updated with patient 12 data where $T^e = 200$. Here the schedule is updated where $T^c = 200$, $T = 280$, $T_1 = 410$ and $T_1 = 560$. The data is then fed into the IP model and we have a new schedule as shown in Table 4.9.

We can see in Table 4.9 that Patient 12 is scheduled for surgery in OT 1 right after Patient 4 has finished. With the addition of Patient 12, the surgery for Patient 3 will be delayed since the IP model has decided that the cost of delay is lower than the cost of overtime. We can see that the arrival of Patient 12

TABLE 4.9: New Schedule after patient 12 arrived

Patient	**OT**	r_i	**Planning Duration**		w_i	**Actual Duration**		**Slot 1**	**Time start**
10	**1**	**0**	**30**	**1000**	**500**	**130**	**1000**	**Finish**	
11	**1**	**0**	**50**	**50**	**60**	**60**	**50**	**Finish**	
4	1	0	50	990	550	90	990	Continue	200min
12	1	10	70	200	60	80	200	New	
2	1	0	130	800	20	140	800		
3	1	0	60	560	15	70	560	Delay	Delay
1	1	0	80	1000	10	100	1000	Delay	Delay
Total			**250**						
7	**2**	**0**	**1000**	**90**	**40**	**1000**	**90**	**Finish**	
8	**2**	**0**	**710**	**60**	**500**	**710**	**100**	**Finish**	
5	2	0	100	85	500	120	80	Continue	200min
6	2	0	500	100	15	500	130		
9	2	0	100	80	15	90	100		
Total				265					

increases the weight of patient 4 and Patient 5 are still undergoing surgery, to prevent interruption.

TABLE 4.10: Schedule at 400 min

Patient	**OT**	r_i	**Planning Duration**		w_i	**Actual Duration**		**Slot 1**	**Time start**
10	**1**	**0**	**30**	**1000**	**500**	**130**	**1000**	**Finish**	
11	**1**	**0**	**50**	**50**	**60**	**60**	**50**	**Finish**	
4	**1**	**0**	**10**	**990**	**500**	**90**	**990**	**Finish**	
12	**1**	**0**	**70**	**200**	**60**	**80**	**200**	**Finish**	
2	1	0	90	800	20	140	800	Continue	400min
3	1	0	60	560	15	70	560	Delay	Delay
1	1	0	80	1000	10	100	1000	Delay	Delay
Total			90						
7	**2**	**0**	**1000**	**90**	**40**	**1000**	**90**	**Finish**	
8	**2**	**0**	**710**	**60**	**500**	**710**	**100**	**Finish**	
5	**2**	**0**	**100**	**85**	**20**	**120**	**80**	**Finish**	
6	**2**	**0**	**500**	**10**	**500**	**500**	**130**	**Finish**	
9	2	0	100	80	15	90	100	Start	400min
Total				80					

The surgeries go on as scheduled until the time where a surgery finishes where T^c, T, T_1, and T_2 are reset and the next surgery started as schedule. For example, in Table 4.10 at 400 minutes, Patient 6 surgery finishes and the schedule is updated where $T^c = 400$, $T = 80$, $T_1 = 150$ and $T_1 = 300$. Without the arrival of any more emergency patients, the final schedule of the day is shown in Table 4.11 where the total duration for both OTs is 500 minutes with two patients delayed to the next day, which are Patient 3 and Patient 1.

TABLE 4.11: After 400 min

Patient	**OT**	r_i	**Planning Duration**		w_i	**Actual Duration**		**Slot 1**	**Time start**
10	**1**	**0**	**30**	**1000**	**500**	**130**	**1000**	**Finish**	
11	**1**	**0**	**50**	**50**	**60**	**60**	**50**	**Finish**	
4	**1**	**0**	**10**	**990**	**500**	**90**	**990**	**Finish**	
12	**1**	**0**	**70**	**200**	**60**	**80**	**200**	**Finish**	
2	**1**	**0**	**90**	**800**	**20**	**140**	**800**	**Finish**	
3	1	0	60	560	15	70	560	Delay	Delay
1	1	0	80	1000	10	100	1000	Delay	Delay
Total						500			
7	**2**	**0**	**1000**	**90**	**40**	**1000**	**90**	**Finish**	
8	**2**	**0**	**710**	**60**	**500**	**710**	**100**	**Finish**	
5	**2**	**0**	**100**	**85**	**20**	**120**	**80**	**Finish**	
6	**2**	**0**	**500**	**10**	**500**	**500**	**130**	**Finish**	
9	**2**	**0**	**100**	**80**	**15**	**90**	**100**	**Finish**	
Total							500		

4.6 Example using Generated Data

4.6.1 Data Generation Procedure

In order to test our ZOP model, we run a computational test with randomly-generated data that test the robustness of our model in real life situation. This is because the number of patients for each day and duration of their surgeries are different. In addition to this, emergency patients arrive randomly and with different number each day. The generated data will simulate the variations between planning durations and actual durations of surgeries. We will generate 10 sets of data for each group of OTs being considered. The data generation procedure can be summarised as follows:

1. Planned processing/operation times for elective patients i:

Generate a planned operation time t^p from the uniform distribution defined on $U[30, 120]$ for $j = 1, \ldots, m$ with the following methods:

- generate a random number R from 1,2,3
- if $R = 1$, set $t^p_{ji} = t^p$
- if $R = 2$, set $t^p_{ji} = t^p + t^e$ where t^e is generated from the uniform distribution $U[10, 40]$
- if $R = 3$, set $t^p_{ji} = 1000$

If there is no j for which $t^p_{ji} = t^p$, randomly select j from $1, \ldots, m$ and reset $t^p_{ji} = t^p$.

Keep generating patients i until the new value of t^p is such that the sum of all t^p values exceeds $480m$–E, where $E = 112.5m$ is the expected planned operation

time for emergency patients (see below), in which case the process terminates without the final patient i. We let E be the expected duration of emergency surgeries. It is equal to the average processing time times 75 multiplied by $1.5m$ (75 is the mean of $U \sim [30, 120]$). We set the limit on booking (total duration time) to be $480m$–E or some multiple of E.

2. Emergency patients:

- Generate the number of emergencies to be an integer from the uniform distribution defined on $U[m, 2m]$.
- Then generate the emergency patients with same processing time distribution as elective patients.
- Generate arrival times for emergency patients from the uniform distribution defined on $U[1, 480]$.

3. Weight for patients:

Weight for patient $i : w_i$ is generated as follows:

- Elective patients: w_i is an integer from the uniform distribution defined on $U[20, 80]$
- Emergency patients: $w_i = 10{,}000$

4. Actual processing/operation times for elective patients i:

Actual operation times for all i and j, $t_{j,i}$ are generated from the uniform distribution on $U[t^p_{j,i} - 20, t^p_{j,i} + 20]$. We set $T = 480, T1 = 540$ and $T2 = 600$.

5. Penalties:

We set the penalties into two bands, where the first is $\beta_{j1} = 50$ and the second is $\beta_{j2} = 200$ for each overtime band being used. The values are the same for all theatres.

4.6.2 Computational Test

To test the data, we change the data generation procedure. We want to see if the different data types will affect the outcome and decide on the following data:

Data Types:

- Original data generation procedure
- High number of emergency patients
- Low number of emergency patients

The outputs we produce are as follows:

Output Produced:

- Number of patients treated
- Number of patients untreated
- Total cost
- Time (second)

4.7 Computational Results

We present the computational results of our methods using the generated data. We generate 10 data and present the results of each data and the average of all

the data. We then change the parameter of the data generation procedure to reflect different situation that might occurs. We generate data to reflect increase and decrease in the number of emergency patients, and increase flexibility of the OTs, where patients can be scheduled into more OTs (the number OTs that are incompatible is reduced).

4.7.1 Results using Generated Data

TABLE 4.12: 2 OTs

Data	**Number of Patients Treated**	**Number of Patients Untreated**	**Total Cost**	**Time (sec)**
1	13	0	100	0.33
2	14	0	100	0.24
3	12	0	0	0.36
4	10	0	50	0.33
5	11	1	30	0.55
6	10	1	40	0.55
7	11	0	200	0.49
8	10	0	250	0.50
9	11	0	0	0.44
10	10	2	170	0.48
Average	**11.2**	**0.4**	**94**	**0.43**

From Table 4.12, there are around 11 patients on average across the data sets. We can see that only in three data sets that there are untreated patients. The total cost suggest that instead of rescheduling, the IP model decided to use overtime to perform surgery. The average time is 0.44 seconds which suggest that the IP model can be calculated in a very short amount of time.

From Table 4.13, there are around 23 patients on average across the data sets. We can see that all data sets no patients were untreated. The is one data

TABLE 4.13: 4 OTs

Data	**Number of Patients Treated**	**Number of Patients Untreated**	**Total Cost**	**Time (sec)**
1	25	0	0	0.39
2	22	0	0	0.39
3	23	0	50	0.39
4	22	0	50	0.39
5	20	0	0	0.23
6	25	0	0	0.49
7	22	0	50	0.27
8	23	0	50	0.91
9	25	0	50	0.44
10	22	0	300	0.42
Average	**22.9**	**0**	**55**	**0.43**

set which has a high total cost. The average time is 0.43 seconds which suggest that even with more data the IP model works efficiently.

TABLE 4.14: 6 OTs

Data	**Number of Patients Treated**	**Number of Patients Untreated**	**Total Cost**	**Time (sec)**
1	36	0	200	0.48
2	35	0	100	0.48
3	32	0	50	0.68
4	35	0	0	0.19
5	31	3	2140	0.65
6	31	3	2140	0.25
7	32	2	550	0.45
8	33	0	50	0.5
9	33	0	50	0.36
10	34	0	0	0.34
Average	**33.2**	**0.8**	**528**	**0.44**

From Table 4.14, there are around 33 patients on average across the data

sets. In one data set the number of patients is 36. We can see that in three data sets that there are untreated patients while the other data sets, all patients were managed to be treated. The total cost is relatively high and in two data sets the cost is 2140. The average time is 0.44 seconds which suggest that the IP model performance does not diminished even with bigger data sets.

4.8 Conclusion

As expected, when there are more OTs, the cost will also increase and that the output produced will also increased. However, our Zero-One-Programming model manages to schedule and reschedule patients efficiently.

Chapter 5

Conclusion and Future Plan

5.1 Concluding Remarks

Our research looks at Combinatorial Optimization Problem in scheduling patients in the OT based on expected operation time using stochastic operation time when the patient arrive into the system. Besides that, we consider online problem on OT where new patients and emergency patients arrive into the system all the time.

We cannot predict the type of patients that will arrive into the system and their operation time are not known precisely in advanced. We design the models and developed the models with suitable data.

We generate the variation of data based on our problem when there is a high number of emergency patient coming into the system and operation time are longer than expected which can lead to disruption of previously booked patients and necessary reschedule is needed. Our research are novel because the models that we developed consider two cost, patient cost and OT cost which are difficult to incorporates into our model. These costs play an important role in producing the schedule because sometimes going into overtime use of OTs might incurred less

cost than scheduling patients to a different day and vice versa, Besides that, we try to avoid repeated delay and cancellation of patients are not allowed. We achieved this by introducing patient's priority and these priority will change according to the patient's conditions. We try to schedule patients as soon as possible based on their priority, where high priority patients will have precedent over other patients.

As the starting point, we schedule patients using simple heuristic technique and develop local search procedure. However, local search procedure does not produce any major improvement because our initial solution was reasonably good. We proposed simulating annealing procedure to improve rescheduling process and the results show an improvement in total cost, cost for delay cases and cost of OT overtime compared with local search and simple heuristic technique. We proposed both the traditional and updated simulated annealing method where in the traditional method, patients are chosen at random to be swap while in the updated version we go through the list in succession. Our variations of data sets that we generate are good because nearly 96 percent achieved good efficiency of the surgery schedule.

In the problem of multiple OTs scheduling in a day, we considered the schedule on the day with the patients booked, emergencies arriving and variations from projected operation duration. The schedule on the day includes the order in which patients are treated, and if the emergency arrivals or variations in operation duration takes us over the theatre slot length the cost of either (or both) of overtime or cancelling operations will be calculated. We actually consider if rescheduling is required after every surgery because of longer duration than planned or there are emergency patients that need surgery immediately. If rescheduling is required, some patients might be moved to different OTs or cancelled on that day. Our proposed model is able to capture both the arrival of emergency and the variation in planning duration and accomodate the schedule accordingly.

In general, the order in which patients are treated in multiple OTs do not influenced the outcome greatly, but it seems that sorting by weight/duration seems to generally do best in terms of cost without reducing the utilisation of the theatres. More data variation is needed to produce a better conclusion.

5.2 Future Work

Build a new algorithm

We will test a new algorithm with difficult parameters included in the model. In this problem, the possible parameters we can consider are value of cost penalty, range of data, and reasonable time for each patient after they are referred into the system. We will:

1. Fixing the time from a week to 6 weeks for the emergency patient or regular patient to be book into the slot in the OT.
2. Looking at different initial schedule and more complex cases for reschedule. This require some changes to the parameters in the algorithm.
3. Looking at how much capacity should we allocate to regular patients and emergency patients. We will consider time or some space in the OT slots for booking emergency patient.
4. Exploring one of the parameter setting in the algorithm, like waiting time of every patient after they come into the system using the same model.
5. Include the surgeon/doctor background experience to model the timing of the duration as another variable.

Disruption Cases

1. Generate different type of emergency situation and other real life scenarios. For example, surgeon cancel operation slot, patient does not show up, system breakdown and other possible disruption situation. We plan to see how several version (parameter) models using the same data and how we rearrange a new schedule.

Bibliography

Aarts, E. and Lenstra, J. K., editors. *Local Search in Combinatorial Optimization*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1997. ISBN 0471948225.

Ahmadi-Javid, A., Jalali, Z., and Klassen, K. J. Outpatient appointment systems in healthcare: A review of optimization studies. *European Journal of Operational Research*, 258(1):3 – 34, 2017. ISSN 0377-2217.

Ahuja, R., Orlin, J., and Sharma, D. Very large-scale neighborhood search. *International Transactions in Operational Research*, 7(4-5):301–317, 2000. ISSN 1475-3995.

Alharkan, I. M. Algorithms for sequencing and scheduling. Available at `http://faculty.ksu.edu.sa/ialharkan/IE428/Algorithms_for_Sequencing_and_Scheduling1.pdf`, 2005. Accessed: 2017-02-17.

Anderson, E. J. and Potts, C. N. Online scheduling of a single machine to minimize total weighted completion time. *Mathematics of Operations Research*, 29(3): 686–697, 2004.

Augusto, V., Xie, X., and Perdomo, V. Operating theatre scheduling with patient recovery in both operating rooms and recovery beds. *Computers & Industrial Engineering*, 58(2):231–238, 2010.

Ausiello, G. *Complexity and Approximability Properties: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 1999.

Azadeh, a., Hosseinabadi Farahani, M., Torabzadeh, S., and Baghersad, M. Scheduling prioritized patients in emergency department laboratories. *Computer methods and programs in biomedicine*, 117(2):61–70, 2014. ISSN 1872-7565.

Bard, J. F. and Purnomo, H. W. Preference scheduling for nurses using column generation. *European Journal of Operational Research*, 164(2):510–534, 2005.

Basson, M. D., Butler, T. W., and Verma, H. Predicting patient nonappearance for surgery as a scheduling strategy to optimize operating room utilization in a veterans' administration hospital. *Anesthesiology*, 104:826–834, 2006. ISSN 0039-6206.

Beliën, J. Exact and heuristic methodologies for scheduling in hospitals: problems, formulations and algorithms. *4OR*, 5(2):157–160, 2007.

Belouadah, H. and Potts, C. Scheduling identical parallel machines to minimize total weighted completion time. *Discrete Applied Mathematics*, 48(3):201–218, 1994. ISSN 0166218X.

Bhattacharyya, T., Vrahas, M. S., Morrison, S. M., Kim, E., Wiklund, R. a., Smith, R. M., and Rubash, H. E. The value of the dedicated orthopaedic trauma operating room. *The Journal of trauma*, 60(6):1336–40, 2006. ISSN 0022-5282.

Blum, C. Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 2(4):353 – 373, 2005. ISSN 1571-0645.

Boldy, D. A review of the application of mathematical programming to tactical and strategic health and social services problems. *Operational Research Quarterly*, pages 439–448, 1976.

Bowers, J. and Mould, G. Concentration and the variability of orthopaedic demand. *Journal of the Operational Research Society*, 53(2):203–210, 2002. ISSN 1476-9360.

Bowers, J. and Mould, G. Managing uncertainty in orthopaedic trauma theatres. *European Journal of Operational Research*, 154(3):599–608, 2004. ISSN 03772217.

Boyar, J. and Favrholdt, L. M. The relative worst order ratio for on-line bin packing algorithms. Technical report, 2003.

Burke, E. K. and Kendall, G. *Search methodologies: introductory tutorials in optimization and decision support techniques*. Springer, 2005.

Calichman, M. V. Creating an optimal operating room schedule. *AORN journal*, 81(3):580–588, 2005.

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

Carter, M. W. and Lapierre, S. D. Scheduling emergency room physicians. *Health care management science*, 4:347–360, 2001. ISSN 1386-9620.

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

Cerny, V. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, 1985. ISSN 00223239.

Cheang, B., Li, H., Lim, A., and Rodrigues, B. Nurse rostering problems–a bibliographic survey. *European Journal of Operational Research*, 151(3):447 – 460, 2003. ISSN 0377-2217.

Clausen, J., Larsen, A., Larsen, J., and Rezanova, N. J. Disruption management in the airline industry-concepts, models and methods. *Computers & Operations Research*, 37(5):809 – 821, 2010. ISSN 0305-0548. Disruption Management.

Cook, S. P vs NP Problem. `http://www.claymath.org/millennium-problems/p-vs-np-problem`, 2017. [Online; accessed 06-12-2017].

Crama, Y., Kolen, A., and Pesch, E. Local search in combinatorial optimization. In Braspenning, P., Thuijsman, F., and Weijters, A., editors, *Artificial Neural Networks*, volume 931 of *Lecture Notes in Computer Science*, pages 157–174. Springer Berlin Heidelberg, 1995. ISBN 978-3-540-59488-8.

Dexter, F. A strategy to decide whether to move the last case of the day in an operating room to another empty operating room to decrease overtime labor costs. *Anesthesia & Analgesia*, 91(4):925–928, 2000.

Dexter, F. and Traub, R. D. How to schedule elective surgical cases into specific operating rooms to maximize the efficiency of use of operating room time. *Anesthesia and analgesia*, 94:933–942, 2002. ISSN 0003-2999.

Dorigo, M. and Blum, C. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2):243 – 278, 2005. ISSN 0304-3975.

Eglese, R. Simulated annealing: A tool for operational research. *European Journal of Operational Research*, 46(3):271 – 281, 1990. ISSN 0377-2217.

Eiben, A. E. and Smith, J. E. *What is an Evolutionary Algorithm?*, pages 15–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. ISBN 978-3-662-05094-1.

Erdem, E., Qu, X., and Shi, J. Rescheduling of elective patients upon the arrival of emergency patients. *Decision Support Systems*, 54:551–563, 2012. ISSN 01679236.

Ernst, E. A., Hoppel, C. L., Lorig, J. L., and Danielson, R. A. Operating room scheduling by computer. *Anesthesia & Analgesia*, 56(6):831–835, 1977.

Fei, H., Meskens, N., and Chu, C. A planning and scheduling problem for an operating theatre using an open scheduling strategy. *Computers & Industrial Engineering*, 58(2):221–230, 2010a.

Fei, H., Meskens, N., and Chu, C. A planning and scheduling problem for an operating theatre using an open scheduling strategy. *Computers & Industrial Engineering*, 58(2):221–230, 2010b.

Gendreau, M., Ferland, J., Gendron, B., Hail, N., Jaumard, B., Lapierre, S., Pesant, G., and Soriano, P. Physician Scheduling in Emergency Rooms. *Practice and Theory of Automated Timetabling VI*, pages 53–66, 2007. ISSN 03029743.

Gerchak, Y., Gupta, D., and Henig, M. Reservation planning for elective surgery under uncertain demand for emergency surgery. *Management Science*, 42(3): 321–334, 1996. ISSN 0025-1909.

Ghazalbash, S., Sepehri, M. M., Shadpour, P., and Atighehchian, A. Operating room scheduling in teaching hospitals. *Advances in Operations Research*, 2012, 2012.

Girotto, J. A., Koltz, P. F., and Drugas, G. Optimizing your operating room: Or, why large, traditional hospitals don't work. *International Journal of Surgery*, 8 (5):359–367, 2010.

Graham, R., Lawler, E., Lenstra, J., and Kan, A. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5(Supplement C):287 – 326, 1979. ISSN 0167-5060.

Grotschel, M. and Lovász, L. Combinatorial optimization. *Handbook of combinatorics*, 2:1541–1597, 1995.

Guinet, A. and Chaabane, S. Operating theatre planning. *International Journal of Production Economics*, 85(1):69–81, 2003.

Gunawan, A. and Lau, H. C. Master physician scheduling problem. *Journal of the Operational Research Society*, 64(3):410–425, 2013. ISSN 0160-5682.

Gupta, D. Surgical suites' operations management. *Production and Operations Management*, 16(6):689–700, 2007. ISSN 1937-5956.

Hall, N. G. and Potts, C. N. Rescheduling for new orders. *Operations Research*, 52(3):440–453, 2004.

Hall, N. G., Potts, C. N., and Sriskandarajah, C. Parallel machine scheduling with a common server. *European Journal of Operational Research*, 136:512–527, 2002. ISSN 0166218X.

Hans, E. W. and Vanberkel, P. T. *Handbook of Healthcare System Scheduling*. Springer US, Boston, MA, 2012.

Herrmann, J. W. *Handbook of production scheduling*, volume 89. Springer Science & Business Media, 2006.

Hertz, A., Taillard, E., and Werra, D. D. A tutorial on tabu search. Technical report, EPFL, Départment de Mathétiques, MA-Ecublens, CH–1015, 1995.

HFMA. Achieving operating room efficiency through process integration. Technical report, Healthcare Financial Management Association, 2005.

Ho, C.-J. and Lau, H.-S. Minimizing total cost in scheduling outpatient appointments. *Management Science*, 38(12):1750–1764, 1992.

Hosseini, N., Sir, M., Jankowski, C., and Pasupathy, K. Surgical duration estimation via data mining and predictive modeling: A case study. *AMIA Annual Symposium Proceedings*, 2015:640–648, 11 2015.

Huang, Y.-L., Hancock, W. M., and Herrin, G. D. An alternative outpatient scheduling system: Improving the outpatient experience. *IIE Transactions on Healthcare Systems Engineering*, 2(2):97–111, 2012.

Jackson, J. R. Scheduling a production line to minimize maximum tardiness. Technical report, University of California, Los Angeles, 1955.

Jackson, J. R. An extension of johnson's results on job idt scheduling. *Naval Research Logistics Quarterly*, 3(3):201–203, 1956. ISSN 1931-9193.

Jebali, A., Hadj Alouane, A. B., and Ladet, P. Operating rooms scheduling. *International Journal of Production Economics*, 99(1):52–62, 2006.

Jebali, A., Ladet, P., and Alouane, H. A. Vers un outil d'aide à la planification et à l'ordonnancement des ressources dans les services de soins. *Th: Institut National Polytechnique de Grenoble*, 2004.

Kelly, J. P. *Meta-Heuristics: Theory and Applications*. Kluwer Academic Publishers, Norwell, MA, USA, 1996. ISBN 0792397002.

Kim, D. W., Na, D. G., and Chen, F. F. Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective. *Robotics and Computer-Integrated Manufacturing*, 19(1-2):173–181, 2003. ISSN 07365845.

Kirkpatrick, S., Gelatt, C. D., Jr, and Vecchi, M. P. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. ISSN 0036-8075.

Klassen, K. J. and Rohleder, T. R. Scheduling outpatient appointments in a dynamic environment. *Journal of Operations Management*, 14:83–101, 1996. ISSN 02726963.

Krempels, K.-h. and Panchenko, A. An approach for automated surgery scheduling. *Proceedings of the Sixth International Conference on the Practice and Theory of Automated Timetabling*, 2006.

Lamiri, M., Xie, X., Dolgui, A., and Grimaud, F. A stochastic model for operating room planning with elective and emergency demand for surgery. *European Journal of Operational Research*, 185:1026–1037, 2008. ISSN 03772217.

Lawler, E. L., Lenstra, J. K., Kan, A. H. R., and Shmoys, D. B. Chapter 9 sequencing and scheduling: Algorithms and complexity. In *Logistics of Production and Inventory*, volume 4 of *Handbooks in Operations Research and Management Science*, pages 445 – 522. Elsevier, 1993.

Leeuwen, J. V. *Handbook of Theoretical Computer Science*. Elsevier Science Inc., New York, NY, USA, 1990. ISBN 0444880755.

Leung, J., editor. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapman & Hall/CRC Computer and Information Science Series. Chapman & Hall/CRC Press, 2004. ISBN 9780203489802.

Liaw, C.-F., Lin, Y.-K., Cheng, C.-Y., and Chen, M. Scheduling unrelated parallel machines to minimize total weighted tardiness. *Computers & Operations Research*, 30(12):1777 – 1789, 2003. ISSN 0305-0548.

Litvak, E. and Long, M. C. Cost and quality under managed care: Irreconcilable differences. *Am J Manag Care*, 6(3):305–12, 2000.

Liu, M., Zheng, F., Chu, C., and Xu, Y. Optimal algorithms for online scheduling on parallel machines to minimize the makespan with a periodic availability constraint. *Theoretical Computer Science*, 412(39):5225–5231, 2011.

Lorber, J. Good patients and problem patients: Conformity and deviance in a general hospital. *Journal of Health and Social Behavior*, pages 213–225, 1975.

Lu, X., Sitters, R., and Stougie, L. A class of on-line scheduling algorithms to minimize total completion time. *Operations Research Letters*, 31(3):232–236, 2003.

Magerlein, J. M. and Martin, J. B. Surgical demand scheduling: a review. *Health services research*, 13(4):418, 1978.

Mancilla, C. *Stochastic Scheduling in Operating Rooms*. PhD thesis, Lehigh University, 2011.

Maniezzo, V., Gambardella, L. M., and de Luigi, F. *Ant Colony Optimization*, pages 101–121. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-39930-8.

Martí, R. and Reinelt, G. *The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization*. Springer Publishing Company, Incorporated, 1st edition, 2011. ISBN 3642167284, 9783642167287.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. Combinatorial minimization. *J. Chem. Phys*, 21:1087–1092, 1953.

Michael, R. G. and Johnson, D. S. Computers and intractability: A guide to the theory of np-completeness. *WH Freeman & Co., San Francisco*, 1979.

Min, D. and Yih, Y. An elective surgery scheduling problem considering patient priority. *Computers and Operations Research*, 37:1091–1099, 2010a. ISSN 03050548.

Min, D. and Yih, Y. An elective surgery scheduling problem considering patient priority. *Computers and Operations Research*, 37(6):1091–1099, 2010b. ISSN 03050548.

Morton, T. and Pentico, D. W. *Heuristic scheduling systems: with applications to production systems and project management*, volume 3. John Wiley & Sons, 1993.

Moz, M. and Pato, M. A genetic algorithm approach to a nurse rerostering problem. *Computers & Operations Research*, 34(3):667–691, 2007.

Moz, M. and Pato, M. V. An integer multicommodity flow model applied to the rerostering of nurse schedules. *Annals of Operations Research*, 119(1-4):285–301, 2003.

Moz, M. and Pato, M. V. Solving the problem of rerostering nurse schedules with hard constraints: new multicommodity flow models. *Annals of Operations Research*, 128(1-4):179–197, 2004.

Narayanaswami, S. and Rangaraj, N. Scheduling and rescheduling of railway operations: A review and expository analysis. *Technology Operation Management*, 2(2):102–122, Dec 2011. ISSN 2249-2364.

Neumann, F. and Witt, C. *Combinatorial Optimization and Computational Complexity*, pages 9–19. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-16544-3.

Niño-Mora, J. Stochastic scheduling. In Floudas, C. A. and Pardalos, P. M., editors, *Encyclopedia of Optimization*, pages 3818–3824. Springer US, Boston, MA, 2009. ISBN 978-0-387-74759-0.

Pardalos, P. M. and Resende, M. G., editors. *Handbook of applied optimization*. Oxford University Press, 2002.

Paschos, V. *Applications of Combinatorial Optimization*. ISTE. Wiley, 2013. ISBN 9781118600344.

Pierskalla, W. P. and Brailer, D. J. Applications of operations research in health care delivery. *Handbooks in operations research and management science*, 6: 469–505, 1994.

Pirlot, M. General local search methods. *European journal of operational research*, 92(3):493–511, 1996.

Pisinger, D. and Ropke, S. *Large Neighborhood Search*, pages 399–419. Springer US, Boston, MA, 2010. ISBN 978-1-4419-1665-5.

Potts, C. N. and Strusevich, V. A. Fifty years of scheduling: a survey of milestones. *Journal of the Operational Research Society*, pages S41–S68, 2009.

Powell, N. H. *Simulation and optimization of healthcare workforce need*. PhD thesis, University of Southampton, 2006.

Pradenas, L., Vidal, F., Parada, L., and Melgarejo, E. An algorithm to surgery scheduling and surgeons assignment in a public hospital. *Revista del Instituto Chileno de Investigación de Operaciones*, 2(1):20–29, 2012.

Pruhs, K., Sgall, J., and Torng, E. Online scheduling. *Handbook of scheduling: algorithms, models, and performance analysis*, pages 15–1, 2004.

Rayward-Smith, V., Osman, I., Reeves, C., and D. Simth, G., editors. *Modern Heuristic Search Methods*. John Wiley & Sons Ltd, 1996.

Schmid, V. and Doerner, K. F. Ambulance location and relocation problems with time-dependent travel times. *European Journal of Operational Research*, 207 (3):1293 – 1303, 2010. ISSN 0377-2217.

Selman, B., Levesque, H. J., Mitchell, D. G., et al. A new method for solving hard satisfiability problems. In *AAAI*, volume 92, pages 440–446, 1992.

Shim, S. O. and Kim, Y. D. Scheduling on parallel identical machines to minimize total tardiness. *European Journal of Operational Research*, 177(1):135–146, 2007. ISSN 03772217.

Sier, D., Tobin, P., and McGurk, C. Scheduling surgical procedures. *Journal of the Operational Research Society*, 48(9):884–891, 1997. ISSN 0160-5682.

Smith, W. E. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2):59–66, 1956. ISSN 1931-9193.

Smith-Daniels, V. L., Schweikhart, S. B., and Smith-Daniels, D. E. Capacity management in health care services: Review and future research directions*. *Decision Sciences*, 19(4):889–919, 1988.

Tao, J., Chao, Z., Xi, Y., and Tao, Y. An optimal semi-online algorithm for a single machine scheduling problem with bounded processing time. *Information Processing Letters*, 110(8):325–330, 2010.

Van Oostrum, J. M., Van Houdenhoven, M., Hurink, J. L., Hans, E. W., Wullink, G., and Kazemier, G. A master surgical scheduling approach for cyclic scheduling in operating room departments. *OR spectrum*, 30(2):355–374, 2008.

Vieira, G. E., Herrmann, J. W., and Lin, E. Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *J. of Scheduling*, 6(1):39–62, January 2003. ISSN 1094-6136.

Wachtel, R. E. and Dexter, F. Influence of the operating room schedule on tardiness from scheduled start times. *Anesthesia and Analgesia*, 108:1889–1901, 2009. ISSN 00032999.

Weinbroum, A. A., Ekstein, P., and Ezri, T. Efficiency of the operating room suite. *American Journal of Surgery*, 185:244–250, 2003. ISSN 00029610.

Williamson, D. P. and Shmoys, D. B. *The Design of Approximation Algorithms*. Cambridge University Press, New York, NY, USA, 1st edition, 2011. ISBN 0521195276, 9780521195270.

Wolsey, L. and Nemhauser, G. *Integer and Combinatorial Optimization*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1999. ISBN 9780471359432.

Wullink, G., Van Houdenhoven, M., Hans, E. W., Van Oostrum, J. M., Van Der Lans, M., and Kazemier, G. Closing emergency operating rooms improves efficiency. *Journal of Medical Systems*, 31(6):543–546, 2007. ISSN 01485598.

Yang, Y., Sullivan, K. M., Wang, P. P., and Naidu, K. D. Applications of computer simulation in medical scheduling. In *Proceedings of the joint conference on information sciences*, volume 227, 2000.

Yeh, J.-Y. and Lin, W.-S. Using simulation technique and genetic algorithm to improve the quality care of a hospital emergency department. *Expert Systems with Applications*, 32:1073–1083, 2007. ISSN 0957-4174.

Yu, G. *Industrial Applications of Combinatorial Optimization*. Applied Optimization. Springer US, 2013. ISBN 9781475728767.

Zhang, Z. and Xie, X. Simulation-based optimization for surgery appointment scheduling of multiple operating rooms. *IIE Transactions*, page 1–15, 2015. ISSN 0740-817X.

Zhang, Z., Xie, X., and Geng, N. Dynamic surgery assignment of multiple operating rooms with planned surgeon arrival times. *IEEE Transactions on Automation Science and Engineering*, 11(3):680–691, 2014. ISSN 15455955.

Zhao, Z. and Li, X. Scheduling elective surgeries with sequence-dependent setup times to multiple operating rooms using constraint programming. *Operations Research for Health Care*, 3(3):160–167, 2013. ISSN 22116923.