Thesis

Using Variable Neighbourhood Search algorithm for tactical level scheduling of elective surgeries in the operating theatre

by

Xiankai Yang

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfilment of the requirements of the degree of

Master of Science

Department of Supply Chain Management

Asper School of Business

University of Manitoba

Winnipeg

Abstract

Operating scheduling is a crucial part of a hospital management. It is related to both patient satisfaction and hospital performance. This thesis considers elective surgery scheduling problems at the tactical level to determine the number of time slots for different elective surgeries. There are multiple objectives, including minimizing the number of overdue patients, number of patients on the waiting list, number of days patients waiting after expected days and maximizing the hospital revenue and performance. All these objectives are fulfilled under the resource limitation, such as the number of wards, the number of beds, the number of ICU rooms, the number of surgeons, and the number of operating theatres are considered in the model as well. Variable neighbourhood search (VNS) algorithm is used in this research to solve the operating scheduling problem. The VNS algorithm in this thesis is composed of several local search schemes based on different neighbourhood structures. A numerical experiment is performed in the randomly generated data set to evaluate the effectiveness of the proposed algorithm. To improve the generalization so that the algorithm can be used in hospitals of different sizes and for different planning periods, we consider several combinations of the number of surgical departments, surgery types and time horizon. In this way, the result of this research is more practical and robust to solve operating theatre scheduling problems of different sizes of hospitals. By using the VNS algorithm in elective surgery operating room scheduling, we hope to increase patients' satisfaction and reduce the cost of hospitals at the same time.

Acknowledgement

I would first like to thank my advisor, Dr. Yuvraj Gajpal and co-advisor, Dr. S.S.Appadoo sincerely. It would have been impossible for me to complete the thesis without their guidance and patience. Also, thanks to my committee member Dr. Changmin Jiang and Dr. Sandeep Arora for their helpful comments and feedback. I would also thank my families for their support from the beginning to the end. Finally, I want to express gratitude to all people who help me with the thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction on operating room scheduling

With the economic development worldwide, people are calling for a higher quality of healthcare facility than before. The hospital plays an important role in the healthcare area, and operation plays an important role in hospitals. According to past research, operating theatre is the primary source of revenue and cost in a hospital (Zhu et al., 2019). Because of the larger ageing population and longer waiting times, the need for efficient scheduling is getting momentum. Thus, scheduling of operating theatre is getting more important nowadays in academic literature. However, the limited resources in operating room scheduling such as surgeons, nurses, wards, beds and ICU etc. make operating room scheduling complicated. As a result, finding better methods for operating room scheduling is becoming a popular topic in the healthcare area.

1.1.1 Levels of operating room scheduling

Based on past research, we can see that decision-making of operating room scheduling can be divided into three levels (Cardoen et al., 2010). The first level is the strategic level, which covers long-term scheduling, the second level is the tactical level which covers mid-term scheduling, and the third level is the operational level which covers

short-term scheduling. At the strategic level, we design plans for operating theatres over a long-time horizon from one year to five years. Decisions of strategic level are made for predicting and distributing funds to all surgical departments. Hence, the decision at the strategic level can not be easily changed once they are made. Capacity planning, allocation, and case-mix problems are three main problems arising at the strategic level. Compared with the strategic level, decisions in the tactical level are for a shorter time horizon, from one month to three months. Master surgery scheduling (MSS) is one of the most important decision at the tactical level. The MSS is about the distribution of operating theatre open time to surgical units, and it's also the objective of this thesis. In the tactical level decision, the capacity of operating theatre and other resources are known or already decided at the strategic level. Based on the capacity, choosing appropriate patients from the waiting list to accept surgeries is the main objective at the tactical level. The last decision level is the operational level which focuses on allocating specific operating room, starting time and ending time to each surgeon. The time horizon at this level is usually a single day. Advance scheduling and allocation scheduling are two main problems at this level. Advance scheduling aims to decide the operating room and the accurate time period for each surgery. The tactical level scheduling considered in this thesis becomes the input variable for operational level scheduling.

1.1.2 Elective Vs non-elective surgeries

Surgeries can be divided into mainly two types based on the patient's urgent need:

elective surgery and non-elective surgery. Patients who need elective surgeries are not urgent patients, so they don't need immediate treatment. Therefore, elective surgeries can be planned in advance. Elective patients are added to the waiting list after their admission. Hospitals allocate them to different operating theatres according to their urgency, position on the waiting list and other conditions. Compared with elective surgery, patients of non-elective surgery can't wait for their surgeries. They should be treated as soon as possible, so they are also called emergency patients. Nowadays, there are mainly two methods for emergency patients. The first one is setting up an emergency operating room for emergency surgeries, and the second one is the insertion of emergency surgeries into the plan. But the second method requires hospitals to leave some time slots for emergency surgeries (Anjomshoa et al., 2018). In this thesis, we focus on elective surgery, and we consider that hospitals have separate operating theatres for emergency surgeries.

In this thesis, we consider operating theatre scheduling problems under the same scenario as described by Anjomshoa et al. (2018). We regard the department as the basic module of a hospital. Surgeries are classified according to conditions and be allocated to each department. There are several operating theatres in the hospital that have all the equipment to perform all types of surgeries. In this thesis, we consider the scheduling under several different situations. Situations are different in the number of departments, number of surgery types or length of time horizon. Time horizon is the whole span of scheduling, such as 16 days, 28 days or 56 days. A single day consists of the time horizon which is referred as the time period in this thesis. The time period is composed

of sessions for different departments in different time periods. Surgeries are measured by session when we decide the length for them.

1.1.3 Urgency category

Another important concept considered in our model is the urgency category of elective surgeries. The urgency category is classified by hospitals to decide how soon a patient should receive surgery. For example, a patient of urgency category 1 should be treated in a time span of 30 days after the admission. According to Anjomshoa et al. (2018), the urgency category can be divided into three levels. If a patient receives surgery during the allotted time span, then the patient is regarded as an on-time patient. Otherwise, the patient is regarded as an overdue patient. An overdue patient may suffer from extra pain because of the delay of surgery, which affects patients' health and hospitals' reputation or even revenue. Surgeries are classified by similarities as well. We call surgeries which are similar to a surgery type. Each department has several surgery types. Surgeons measure the conditions of all patients according to urgency category and surgery type and put them on the waiting list.

1.1.4 Constraints for operating room scheduling

Besides operating theatre and surgeons, downstream resources are also considered in this thesis. Downstream resources include the number of wards, number of beds and number of ICU units. After each surgery, there is the possibility that a patient has to be

sent to ICU room for further care. Patients who don't need ICU care are sent to the wards, and beds are assigned. After the recovery period, patients are allowed to leave the hospital. The number of days a patient stays in the hospital is called the *length of stay* in this thesis.

1.1.5 Objective function

The operating room scheduling problem considered in this thesis tries to optimize many factors at the same time. Following multiple objectives are considered in this thesis.

1. **Increasing revenue:** The main objective is maximizing the revenue. It is calculated by multiplying the revenue of each surgery type and the number of surgeries. The aim is to maximize the revenue.
2. **Increasing hospital performance:** Hospitals have to complete a certain number of surgeries to achieve a different level of performance. We have a reward point system to measure the performance of the hospital. For example, if the portion of surgeries, which is given to on-time patients, is higher than 65%, then the hospital can get 2 points. The more points the hospital receives, the better performance it has. Different weights are given to levels, and the results should be maximized.
3. **Reducing overdue patients**: Patients are classified by urgency category, and each of them has an expected waiting day before receiving surgery. If a patient doesn't receive the surgery before the date, it is regarded as overdue. We expect to minimize the number of overdue patients at the end of the time horizon.

4. **Reducing patients on the waiting list**: There are overdue patients on the waiting list. If a patient is waiting for a surgery at the end of the time horizon and it is still in the expected waiting time, it is regarded as an on-time patient. At the end of the time horizon, all patients on the waiting list, including on-time patients and overdue patients, are calculated. The number of it should also be minimized.
5. **Reducing overdue patients waiting time:**. Any days after the expected days are considered as overdue patient waiting time, which is measured by time periods. Waiting time longer than expected can harm patients' health conditions, which will lead to a reduction in hospital performance. As a result, this number should also be minimized.

1.2 Introduction on variable neighbourhood search (VNS)

Mladenović & Hansen (1997) firstly proposed variable neighbourhood search (VNS) as a method to solve the combinatorial optimization problem. VNS is a metaheuristic that is a framework for solving combinatorial and global optimization problems (Hansen et al., 2010). Hansen et al. (2010) also pointed out that the basic idea of VNS is changing the neighbourhoods to find a minimum, and after that, perturbation is done so that it can move to the next valley (Hansen et al., 2017).

Hansen et al. (2017) proposed that VNS is composed of three steps which are shaking procedure, improvement procedure (local search) and neighbourhood change step. By using the shaking procedure, we expect to avoid local minimum traps (Hansen et al.,

2017). For example, we have a set of neighbourhood structures as $A_k, 1 \leq k \leq k_{max}$. A solution x is given for k^{th} neighbourhood structure as $A_k(x)$. Basically, in the shaking procedure, we randomly choose a solution from k^{th} neighbourhood. The second step, the neighbourhood change step, is to make the heuristic move to another neighbourhood to find more solutions. Take sequential neighbourhood change as an example, the algorithm keeps improving the solution in one neighbourhood. If there isn't any improvement, the algorithm moves to the next neighbourhood and continues searching. Improvement procedure is the last step. Let's take the set $A_k(S)$ as an neighbourhood set of solution S,. Firstly, an initial solution is randomly picked from the set. Then the algorithm searches for a better solution in the neighbourhood structure for each iteration. If a better solution is found, it replaces the old solution and becomes the new best solution S' . This step repeats until no better solutions can be found in this neighbourhood. This strategy is called the *first movement* (Hansen et al., 2017). This strategy is also applied in this thesis.

Figure 1.1 is a flowchart of the Variable neighbourhood search process. We will explain the VNS working principle based on Figure 1.1. From figure 1.1, we can see that in the beginning, we have a set A and the maximum number of neighbourhood structure is k_{max}. The term k represents the sequence of neighbourhoods. The term S is the initial solution. In the shaking procedure, S' is generated randomly as the best solution in k^{th} neighbourhood. Then, in local search procedure, it completes the search and gets a new solution S''. If S'' is better than S', S'' replaces S' and becomes the best solution. After that, it goes back to the shaking procedure and repeats. However, if S'' is not improved

compared with S', then the algorithm goes to the next neighbourhood and perform the search operation until the last neighbourhood is executed. If it finishes searching the last neighbourhood, it checks whether the stopping condition is fulfilled or not. The process repeats until the stopping condition is fulfilled.

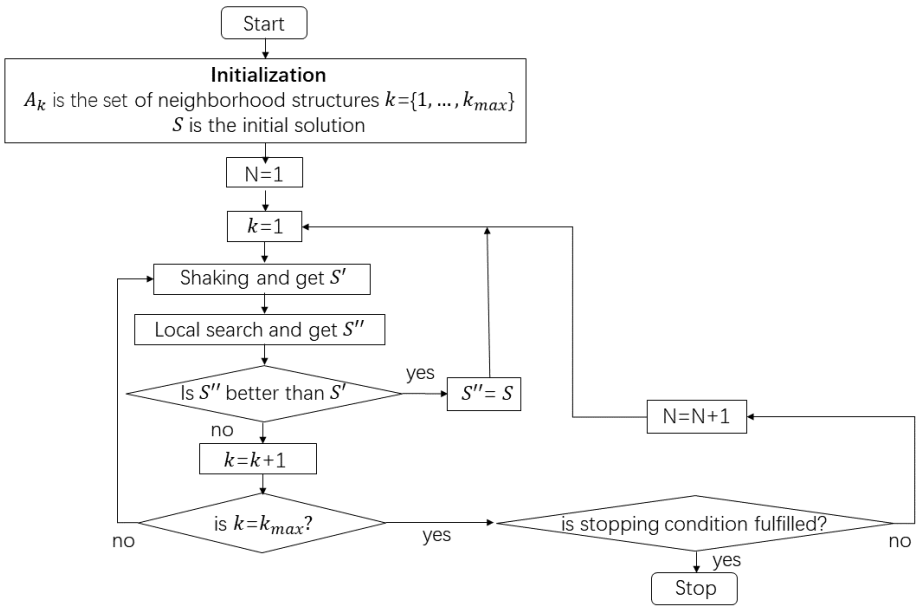

Fig 1.1 Variable Neighbourhood Search

1.3 Contribution and overview of the thesis

The main contribution of this thesis is filling the gap in literature of using metaheuristics to solve operating theatre scheduling problems. When our VNS algorithm is applied to large instances problem, it can give solutions in reasonable running time while CPLEX is not able to solve the problem. We also make sensitivity analyses to check how some constraints and objective functions influence the problem. The results can help hospital managers make decisions in a practical scenario.

In chapter 2, we introduce literatures related to operating scheduling problem, tactical level scheduling and VNS algorithm. Then we give all the notations, decision variables, objective functions and constraints in chapter 3. The problem description is also included in this chapter. Chapter 4 introduces the framework of our VNS algorithm and provide details of all the local searches. Chapter 5 mainly describes all the experiments we conduct by using the solutions given by our VNS algorithm. For the last chapter, we conclude the thesis.

Chapter 2

Literature Review

Scheduling has been an important problem in the industry area for a long time. In recent years, much research has been done to optimize scheduling. For example, two-agent single machine is one of the topics being researched a lot (Gajpal et al., 2014; Li et al., 2021a; Li et al., 2021b; Li et al., 2020a; Li et al., 2020b; Sahu et al., 2018; Li et al., 2018). Besides two-agent single machine problem, much research also aims at using metaheuristic to solve flowshops-related problem (Ali et al., 2017; Ali et al., 2021; Bhardwaj et al., 2020; Gajpal et al., 2006; Galpal & Rajendran, 2006).

According to Cardoen et al. (2010), research about operating room scheduling became popular in late 1990s or even 2000. At first, research focused on the cost in the operating theatre, and it is regarded as a problem only in the healthcare area (Macario et al., 1995). Later, Lagergren (1998) proposed that knowledge of operational management could be highly combined with operating room scheduling and proved that lots of contributions had been made in this way. Hulshof et al. (2012) proposed a taxonomy classification of scheduling in healthcare based on decision level. According to their research, the objective of the strategic level is to create a map for the whole organization. Assignment should be distributed to individuals based on decisions of this level. As a result, strategic-level decisions are designed for a long time horizon and composed of many forecasts. The tactical level is considered as a connection between the strategic level

and operational level. That's why the tactical level is about middle-term decision-making. It translates the strategic-level decisions and guides the operational-level decision-making. Compared with the strategic level, tactical level decisions give instructions on processing each operation in more detail. At last, the operational level is about short-term decision-making. Decisions at this level are based on the operational level. They give exact information about each surgery, like when and where the surgery will be performed. Patients and resources are considered individually at this level. The time horizon in this level is short, so flexibility is less compared with the other two-level.

Because of emergency patients, uncertainty becomes an important factor in an operating theatre scheduling problem. Surgery duration is another factor that is uncertain in operating theatre scheduling. Any unpredictable accident in operating theatre could lead to delay of the subsequent surgeries. Based on this fact, research can be classified as deterministic and stochastic. Any uncertainty during the operating process can interrupt the whole scheduling. To solve this problem, Lebowitz (2003) used Monte Carlo simulation in research. Lamiri et al. (2008) designed a stochastic model by combining Monte Carlo simulation and Mixed Integer Programming. In their research, they consider elective patients and emergency patients at the same time. Pham et al. (2008) generalized the operating room scheduling problem as a job scheduling problem and proposed a mixed-integer linear program to solve it. Van der Lans et al. (2006) compared five different methods on arranging emergency patients to operating theatre scheduling and found that leaving slack in the schedule for all the operating

rooms is the best method.

Like as mentioned before, besides leaving slack in scheduling, hospitals can also leave specific operating rooms for emergency patients. In this way, the uncertain effect of emergency patients could be minimized. Under this condition, operating room scheduling can be regarded as a deterministic problem. Adan & Vissers (2002) used a case study to figure out how to assign patients to surgeons based on resources limitation like the number of ward beds, nurses capacity and intensive care rooms capacity. Testi et al. (2007) proposed a three-phase method to reduce patient waiting time and increase utilization at the same time. By simulating their method in Witness software, the three-phase method was proved to be effective. Bowers & Mould (2005) did their research in an ambulatory care facility that only received elective patients. They also simulated the scenario on the computer, and utilization increased by 18%. Their result can be generalized to hospitals that have separate operating rooms for the emergency patient.

Metaheuristic has become an effective tool to solve scheduling problems such as capacitated green vehicle routing problems, green vehicle routing problem and multi-depot green vehicle routing problem (Zhang et al., 2018; Peng et al., 2019; Zhang et al., 2020). Variable neighbourhood search (VNS) is a metaheuristic that is often used to solve combinational optimization problem. Islam & Gajpal (2021) combined VNS and ant colony optimization in their research to solve vehicle routing problem with time windows. Islam et al. (2021a) combined particle swarm optimization and VNS to solve mixed fleet based green clustered logistics problem. In the other research, they developed a hybrid metaheuristic in the same way to solve clustered vehicle routing

problem (Islam et al., 2021b).

2.1 Operating theatre scheduling at strategic level

Decision level is another main topic in an operating theatre scheduling problem. Choi & Wilhelm (2014) proposed a non-linear programming to solve the capacity planning problem at the strategic level. They aimed to reduce the cost related to penalties from overdue patients. Capacity allocation is another strategic level problem. Capacity allocation is used to assign surgeons to operating rooms in all time periods (Zhu et al., 2019). The objective of the capacity allocation problem is different from the capacity planning problem. Its objective is to minimize the waiting time for patients, which means that it is not cost-effective. Creemers et al. (2012) designed a bulk service queueing model to minimize expected waiting time. The case-mix problem is another main problem at the strategic level. Instead of assigning surgeons to the operating theatre's time period, the case-mix problem aims to assign operating theatre blocks to surgeons. One feature of this problem has been adopted in the tactical level model considered in this thesis – dividing surgeries into several types. According to the research of Lehtonen et al. (2013), accuracy of surgery type classification and duration can help improve the utilization of operating theatres.

2.2 Operating theatre scheduling on the tactical level

As we mentioned before, the problem we have to face at this decision level is Master

Surgery Scheduling (MSS) problem. The basic idea of this problem is to assign time blocks of operating theatres to each surgical unit, which is the department in this thesis. Penn et al. (2017) proposed multiple criteria integer programming to solve this problem. Their research considers many factors that can affect decision-making, such as surgeon's availability, equipment availability and flexibility to change the time horizon, etc. Cardoen et al. (2010) considered MSS problem from another perspective. In their research, the occurrence of medical precautions is an important factor that influences the sequence of operations. Column generation and a dynamic programming algorithm are applied to solve the problem. Beliën et al. (2009) gave a different idea in their paper. They proposed that an operating theatre shouldn't be shared by several surgeons or departments. A mixed-integer programming is used, and a metaheuristic is applied to solve their MSS problem.

In tactical level scheduling, there are many multiple-objective types of research. For example, Testi et al. (2007) proposed a model to reduce overtime, patients on the waiting list and improve utilization. According to Chaabane et al. (2008), they gave a model to reduce cost and maintain the service quality for patients at the same time. Papers can also be divided by the constraints they consider. Chaabane et al. (2008) considered surgeons and operating theatres, which are upstream resources. Fügener et al. (2014) choose ICU as the most important constraint, a downstream resource. Most of the papers concentrate on some of the objectives and constraints. However, Anjomasoa (2018) proposed a model which includes both upstream and downstream resources, and multiple objectives are included in it as well. Besides that, they also

regard the limitation of the base plan, which is determined at the strategic level, as one of the constraints. As we mentioned before, strategic level and tactical scheduling are closely related. Any change of tactical scheduling should be guided by the base plan. As a result, it increases the flexibility of tactical scheduling and makes it more practical when applied to a real scenario.

2.3 Operating theatre scheduling at the operational level

Two main problems are often discussed at the operational level – advance scheduling and allocation scheduling. Advance scheduling is about assigning operations to certain days over the time horizon. Ceshia & Schaerf (2016) used local search in their paper to solve advanced scheduling problems. In their paper, besides operating theatre, the bed is the most important constraint in the decision-making process. An instance generator was also created so that the gap between theory and reality could be narrowed. Jebali & Diabat (2015) designed a stochastic model to solve advanced scheduling problems. They not only consider operating theatres and beds, but also consider ICU rooms as another resource constraint. CPLEX is used to solve their problem.

Allocation scheduling decides the operation start time and distribution of upstream and downstream resources. Upstream resources include surgeons, nurses, operating theatres and equipment, while downstream resources include beds, wards and ICU rooms. Castro & Marques (2015) proposed a two-level generalized disjunctive programming model in their paper. The first level determines the operating theatres for surgeries, and the second level assigns exact surgeons to operating rooms and gives them an exact

time. Decomposition algorithm is used in their paper to provide solutions. A two-stage decision process is also used by Lee &Yih (2014), but it works differently. In this process, the order of all operations is decided firstly and in the second stage starting time of each surgery is determined. They generalize their problem as a flexible job shop model. A combination of genetic algorithm and decision-heuristic is used in two decision stages.

2.4 Operating theatre scheduling at more than one level

Although decision-making level is divided into three levels, more researchers are trying to study operating theatre scheduling at two or three levels at the same time. For example, the model proposed by Guido and Conforti (2017) was designed for both strategic and tactical levels. Aringhieri et al. (2015) mentioned in their paper that three decision levels can not be divided clearly, so they gave a 0-1 linear programming model based on the close relationship between decision making of the tactical and operational level at the same time. A metaheuristic is designed to solve the problem, and they found that there was an inherent hierarchy between two decision levels.

2.5 Variable neighbourhood search on operating theatre scheduling

Although there is much research about operating theatre scheduling, using variable neighbourhood search (VNS) in this area is not very common. In the healthcare delivery area, VNS is just commonly used to solve nurse rostering problems (Delleaert & Jeunet,

2017). Because of the effectiveness, Delleaert & Jeunet (2017) used VNS to solve an operating theatre scheduling problem. They focused on the utilization of operating rooms, beds and nurses at the same time. Multiple objectives make the problem more complex, which is also the reason they choose VNS. A mixed-integer programming model is proposed, and both VNS and CPLEX are used to solve the problem. According to their research, VNS is more effective in solving their problem. The method they use is a variant of VNS called Random VNS. Firstly, it is different from traditional VNS; their shaking procedure occurs only when it finds that local search doesn't improve the best solution. In their paper, patients are allocated to different days and different operating rooms to receive surgery, and they are classified by surgery category. Utilization is calculated by the algorithm based on dates, operating rooms and surgery category. Then their local search finds overutilization among all the solutions. After that, it recombines the date, operating theatre and surgery category to get a better solution. Then the process repeats until they get the best solution.

2.6 Operating theatre scheduling problem considered in this thesis

This thesis uses the same scenario as considered by Anjomshoa et al. (2018). Decisions of scheduling are made based on the tactical level. Only elective surgeries are considered in the thesis. Surgeries are divided into different surgery types. Each surgery type has its urgency category and belongs to a department. After admission, surgeons assess patients and put them on the waiting list. After surgery, patients are sent to the wards or ICU room according to their condition. In this way, meeting downstream

resources is considered an objective as well. Hospitals have a base plan which is about funding, upstream resources capacity and downstream resources capacity. The operating theatre schedule is made based on this base plan. We have multiple objectives such as increasing the revenue, increasing the overall performance, reducing overdue patients, reducing patients on the waiting list and reducing waiting days after patients are overdue. By giving objectives different weights, we can change the importance of each objective so that it can be generalized to more hospitals.

Chapter 3

Problem description

3.1 Operating theatre scheduling problem description

In this thesis, we discuss the operation theatre scheduling at the tactical level, and it is based on decisions at the strategic level, which is a base plan in this thesis. The base plan decides how much funds the hospital will receive in the long term, like one year. Then the number of total sessions for each department is determined based on funds received by the hospital. As mentioned before, changes can be made to the base plan, but the number of changes is limited. As a result, our model has constraints on the numbers of added, deleted and changed sessions during the whole time horizon to accommodate the varying number of surgery types. For example, the base plan allows us to have 100 sessions in the next three months. The maximum number of added and deleted sessions is 5, and the maximum number of changed sessions is 7, which means that we can add or delete 5 sessions from the base plan, but the total number of added and deleted sessions must be less than 7.

After admission, patients are classified by surgery types. Each surgery type has an urgency category, and each surgery belongs to a particular department. The urgency category decides the number of time periods patients can be on the waiting list without being overdue. In this thesis, the urgency categories 1,2 and 3 are due by 30, 90, and

365 days.

Based on the limitation of resources such as surgeons, operating theatres, sessions, beds and ICU rooms, our algorithm decides how many sessions should be allocated to each department and how many patients of each surgery type should be treated on different time periods. A mixed-integer linear programming (MILP) model is applied in this thesis. The MILP model is solved using a CPLEX software. Further the thesis proposes a metaheuristics approach to solve the problem. In the next part, we will list all the notations objectives and constraints.

3.2 Notations and formulas

This paper considers the model proposed by Anjomshoa et al. (2018). We try to stick with the similar notations defined in this article to avoid confusion.

3.2.1 Parameters

The proposed model has the following parameters.

np	The number of time periods.
$C = \{1,2.3\}$	Urgency category for different surgeries.
$L_c , c \in C$	Maximum number of time periods a patient of certain urgency category can stay on the waiting list.
$p^{past} = \{-max_{c\in C} L_c, \dots ,0\}$	Maximum number of time periods from which a patient is sent to the waiting list before the first time

	period of the planning period without being overdue.
nw	The number of wards.
$b_{w,p}$	The number of beds in wards w in time period p.
nd	The number of departments in the hospital.
D_d	The maximum number of sessions which can be allocated to department d in a time period.
$F_{d,p}$	The number of available surgeons for department d in time period p.
ns	The number of surgery types.
$SD_d \subseteq S$	The subset of surgery type S which belong to department D.
NSD_d	The number of surgery types associated with department d.
$SDO_{d,j}$	The j^{th} surgery type associated with department d.
$SC_c \subseteq S$	The subset of surgery type S which belong to urgency category C.
NSC_c	The number of surgery types in urgency category c.
$SCO_{c,j}$	The j^{th} surgery type associated with urgency category c.
τ_s	The duration of surgery type s.
LOS_s	The length of stay in hospital after receiving surgery s.
R_s	The revenue hospital can earn after performing surgery s.

w_s	The ward a patient will be sent to after receiving surgery s.
ICU_s	The probability of surgeries of surgery type s going to ICU room.
$a_{s,p}$	The patient arriving on the waiting list in time period $p \in P$ to receive surgery of surgery type $s \in S$.
$e_{s,p}$	The patient arriving on the waiting list in time period $p \in P^{past}$ to receive surgery of surgery type s.
N^{OT}	The number of operating theatres.
$num\_sessions$	The number of sessions an operating theatre can hold in any time period. The $num\_sessions$ is 2 by default in this thesis.
$session\_length$	The duration of a session, which is 2 hours in this thesis.
N^{ICU}	The capacity of ICU in a time period.
MAX^{SESS}	The maximum number of sessions can be held in the hospital over the whole time horizon.
$N_{d,p}^{SESS}$	The number of sessions which are allocated to department d in time period p. This term is declared in the base plan.
MAX^{add}	The maximum number of sessions which can be added to the base plan over the whole time horizon.
MAX^{del}	The maximum number of sessions which can be deleted from the base plan over the whole time

	horizon.
MAX^{chg}	The maximum number of changes which can be made to the base plan over the whole time horizon.
$C' = \{2,3\}$	The urgency categories that influence the measurement of hospital performance. Surgeries of urgency category 1 must be performed on time in this thesis so it is not included in C'
$J = \{1,2,3\}$	The reward points of achieving different target level. Target level is measured by dividing on time surgeries by all surgeries performed.
$Q = \{1,2,3,4\}$	Four quarters of the whole time horizon.
$\alpha_{c,q,j}\ c \in C', q \in Q, j \in J$	The thresholds of achieving reward point j in quarter q for surgery of urgency category C'.

Table 3.1 Parameters of operating theatre scheduling problem

3.2.2 Decision variables

The proposed model mainly determines the number of sessions assigned to each department at each time period $X_{d,p}$ and the number of patients from each surgery type receiving surgery in each time period $\theta_{s,p}$. Other decision variables depend on these two main decision variables. For clarity of the model, decision variables are divided into four categories: 1) main decision variables, 2) surgery related decision variables, 3) department related decision variables, and 4) quarter related decision variables.

Main Decision variables	
$X_{d,p}$	The number of sessions allocated to department d in period p.
$\theta_{s,p}$	The number of patients receiving surgery of surgery type s in time period p.
Surgery related decision variables	
$\theta_{s,p}^{+}$	The number of on time patients receiving surgery of surgery type s in time period p.
$\theta_{s,p}^{-}$	The number of overdue patients receiving surgery of surgery type s in time period p.
$\Omega_{s,p}$	The number of patients who are overdue at the beginning of time period p and need surgery of surgery type s.
$\Omega_{s,p}^{+}$	The number of on time patients on the waiting list who need surgery of surgery type s at the end of time period p.
$\Omega_{s,p}^{-}$	The number of overdue patients on the waiting list who need surgery of surgery type s at the end of time period p.
Department related decision variables	
$v_{d,p}^{add}$	The number of sessions which are added to department d in time period p.
$v_{d,p}^{del}$	The number of sessions which are deleted from

	department d in time period p.
$v_{d,p}^{chg}$	The number of sessions which are changed from department d in time period p.
Quarter related decision variable	
$\Pi_{c,q}^{+}$	The number of on time patients of category $c \in C'$ who received surgery in quarter $q \in Q$.
$\Pi_{c,q}^{-}$	The number of overdue patients of category $c \in C'$ who received surgery in quarter $q \in Q$.
$Y_{c,q,j}$	If reward point $j \in J$ is achieved in quarter $q \in Q$ by performing surgeries of urgency category $c \in C'$, then $Y_{c,q,j} = 1$. Otherwise, $Y_{c,q,j} = 0$.

Table 3.2 List of decision variables

3.2.3 Objective functions

The objective of the operating theatre scheduling problem is to maximize the net-revenue of hospitals. The net-revenue consists of the generated revenue and the performance reward, minus the penalty incurred due to overdue patients, patients on the waiting list and tardy days of overdue patients. Thus, the objective function has the following five components. We measure the net-revenue by using the weight for revenue, performance reward and penalty for overdue patients, tardy days. The net revenue can be considered as an overall performance of the hospital because the net revenue not only considers the revenue but also considers other factors related to

patients.

$$Z_R = \sum_{s\in S}\sum_{p\in P} R_s \times \theta_{s,p} \quad (1)$$

$$Z_J = \sum_{c\in C\prime}\sum_{j\in J} Y_{c,q,j} \quad (2)$$

$$Z_{\Omega^-} = \sum_{s\in S} \Omega^-_{s,P} \quad (3)$$

$$Z_{\Omega^-,\Omega^+} = \sum_{s\in S}(\Omega^+_{s,P} + \Omega^-_{s,P}) \quad (4)$$

$$Z_T = \sum_{s\in S}\sum_{p\in P} \gamma_s \times \Omega_{s,p} \quad (5)$$

Max $weight_1 \times Z_R + weight_2 \times Z_J - weight_3 \times Z_{\Omega^-} - weight_4 \times Z_{\Omega^-,\Omega^+} - weight_5 \times Z_T$ (6)

Our overall objective is to maximize the net-revenue of the hospital. The objective (1) calculates the total revenue by multiplying revenue for each surgery R_s with all surgeries performed. The second objective, objective (2), is to improve the performance of the hospital. There are thresholds for the hospital to achieve. If the percentage of on-time patients meets the requirement, the hospital gets rewards points, which means that the performance is good. As a result, the value should be big. In this thesis, we divide the whole time horizon into four quarters q. And there is a threshold $\alpha_{c,q,j}$ of different performance levels, to get reward point j for each quarter. Because patients in urgency category 1 are the most urgent patients, no category 1 patient is allowed to be overdue. So only patients of categories 2 and 3 are involved in performance measurement. Thus, categories 2 and 3 are called costed categories in this thesis.

The objective (3) considers the penalty incurred due to overdue patients. Here Z_{Ω^-} is

the number of overdue patients of all surgery types at the end of time horizon, which should be minimized. The objective (4) considers the penalty incurred due to patients on the waiting list. Z_{Ω^-,Ω^+} represents the number of both overdue and on-time patients who still wait for treatment at the end of the time horizon. This number should also be minimized. An overdue patient still needs to receive the surgery as soon as possible because long-time waiting may worsen their health. We want to minimize the number of time periods before the overdue patients get their surgeries. The number of periods starts counting when a patient becomes overdue. As a result, objective (5) is proposed and γ_s is the weight for each surgery type s.

All 5 objectives are included in the objective function (6), which is the net-revenue of the hospital. The purposes of net-revenue are increasing profit, improving performance and providing better service at the same time by minimizing the penalty. The value of net-revenue should be maximized. The penalty and weights can be changed according to the actual demand of the hospital based on the practical scenario.

3.2.4 Constraints

Our model considers many constraints such as maximum session availability, maximum added sessions, maximum deleted sessions, maximum changed sessions, maximum treated patients, maximum beds availability, maximum ICU rooms availability, maximum operating rooms availability and maximum surgeon availability. The equations for different constraints are outlined below.

$$\sum_{d\in D,p\in P} X_{d,p} \le MAX^{SESS} \quad (7)$$

$$X_{d,p} + v_{d,p}^{del} - v_{d,p}^{add} = N_{d,p}^{SESS} \qquad \forall d \in D, p \in P \quad (8)$$

$$\sum_{d\in D,p\in P} v_{d,p}^{add} \le MAX^{add} \quad (9)$$

$$\sum_{d\in D,p\in P} v_{d,p}^{del} \le MAX^{del} \quad (10)$$

$$\sum_{d\in D,p\in P} (v_{d,p}^{add} + v_{d,p}^{del}) \le MAX^{chg} \quad (11)$$

$$\sum_{s\in SD_d} \tau_s \theta_{s,p} \le session_{length} \times X_{d,p} \qquad \forall d \in D, p \in P \quad (12)$$

$$\sum_{s\in SD_d} \tau_s \theta_{s,p} > session_{length} \times (X_{d,p} - 1) \qquad \forall d \in D, p \in P \quad (13)$$

$$\sum_{s\in S} \sum_{p'\in\{0,\dots,p-1\}} \theta_{s,p-p'} \times B_{s,p'} \le b_{w_s,p} \qquad \forall d \in D, p \in P \quad (14)$$

$$\sum_{s\in S} \theta_{s,p} \times ICU_s \le N^{ICU} \qquad \forall p \in P \quad (15)$$

$$\Omega_{s,p}^{-} = \max\left\{0, \sum_{p'\in P^{past}, p-p'\ge L_{c_s}} e_{s,p'} + \sum_{p'\in P, p-p'\ge L_{c_s}} a_{s,p'} - \sum_{p'\in P, p'\le p} \theta_{s,p'}\right\} \quad \forall p \in P, s \in S \quad (16)$$

$$\Omega_{s,p}^{+} = \sum_{p'\in P^{past}} e_{s,p'} + \sum_{p'\in P, p'\le p} a_{s,p'} - \sum_{p'\in P, p'\le p} \theta_{s,p'} - \Omega_{s,p}^{-} \quad (17)$$

$$\Omega_{s,p} = \max\left\{0, \sum_{p'\in P^{past}, p-p'\ge L_{c_s}} e_{s,p'} + \sum_{p'\in P, p-p'\ge L_{c_s}} a_{s,p'} - \sum_{p'\in P, p'<p} \theta_{s,p'}\right\}$$

$$\forall p \in P, s \in S \quad (18)$$

$$\theta_{s,p}^{-} = \Omega_{s,p} - \Omega_{s,p}^{-} \qquad \forall p \in P, s \in S \quad (19)$$

$$\theta_{s,p}^{+} = \theta_{s,p} - \theta_{s,p}^{-} \qquad \forall p \in P, s \in S \quad (20)$$

$$X_{d,p} \le D_d \qquad \forall d \in D, p \in P \quad (21)$$

$$\sum_{d\in D} X_{d,p} \le num_{sessions} \times N^{OT} \qquad \forall p \in P \quad (22)$$

$$X_{d,p} \le F_{d,p} \qquad \forall d \in D, p \in P \quad (23)$$

$$\Omega_{s,p} = 0 \qquad \forall p \in P, s \in S_1 \quad (24)$$

$$\Pi_{c,q}^{+} = \sum_{s\in SC_c} \sum_{p\in P} \theta_{s,p}^{+} \qquad \forall c \in C', q \in Q \quad (25)$$

$$\Pi_{c,q}^{-} = \sum_{s\in SC_c} \sum_{p\in P} \theta_{s,p}^{-} \qquad \forall c \in C', q \in Q \quad (26)$$

If $\Pi_{c,q}^{+} \geq \alpha_{c,q,j}\left(\Pi_{c,q}^{+} + \Pi_{c,q}^{-}\right) \rightarrow Y_{c,q,j} = 1 \qquad \forall c \in C', q \in Q$ (27)

Constraint (7) makes sure that the number of allocated sessions doesn't exceed the maximum sessions specified by the base plan. Constraint (8) bound the variable according to the changes to the base plan. Then the maximum number of added, deleted and changed sessions are limited in constraints (9), (10), (11).

In constraint (12), we calculate the total duration of all surgeries and the product of allocated sessions with session length. By comparing the two results, the surgery duration should not exceed the allocated session time. Constraint (13) makes sure that every session is allocated and there is no waste of sessions.

Constraints (14) and (15) are about downstream resources. They limit the number of patients sent to beds in different wards and ICUs after the surgery.

Constraint (16) calculates the number of overdue patients. Patients who exist on the waiting list before the first time period and those who arrive after the first period can both receive surgeries before the expected date. The sum of them minus the number of patients who receive surgeries before time period p and we will get the result of how many patients are still on the waiting list at the end of p. Constraint (17) calculates the on-time patients on the waiting list at the end of the time period p. Constraint (18) is similar to constraint (16); the difference is that it calculates the number of overdue patients at the beginning, instead of at the end, of time period p. After we get the result of constraints (16), (17) and (18), we can get the number of on-time and overdue patients who receive surgeries in time period p. The calculation is shown in constraints

(19) and (20).

In constraint (21), the number of sessions allocated to departments is limited so that it doesn't exceed the maximum number of sessions the department can accommodate. Constraint (22) gives the limitation of the number of operating theatres. The number of allocated sessions should be less than the number of sessions a hospital can hold per day. Limitations of surgeons is considered in constraint (23). Each session is allocated to a surgeon in such a way that the total allocated sessions shouldn't exceed available surgeons. The objective of constraint (24) is to make sure that all the patients of urgency category 1 must be treated on time. Otherwise, patient's health condition can be harmed, which can affect service quality and hospital performance at the same time.

At last, constraints (25), (26) and (27) are given to calculate $Y_{c,q,j}$, which is described in the objective function part. To calculate $Y_{c,q,j}$ in objective (5), we need to calculate the number of on-time and overdue patients at first, which is $\Pi_{c,q}^{+}$ and $\Pi_{c,q}^{-}$. In constraint (26), we calculate the value of $Y_{c,q,j}$. If the percentage of treated on-time patients is higher than or equal to the threshold $\alpha_{c,q,j}$, then $Y_{c,q,j} = 1$, otherwise, $Y_{c,q,j} = 0$. For example, in quarter 1 and category 2, 70 on-time patients and 30 overdue patients are treated. If the threshold is 70 % to achieve the reward points of 3, then $\alpha_{c,q,j}, = 0.7$, which represents that the percentage of on-time patients is 70%. As a result, $Y_{2,1,3} = 1$. Besides increasing net-revenue and improving hospital performance, we also want to provide patients with better service.

Chapter 4

Variable Neighbourhood Search

4.1 Framework of VNS

The problem considered in this thesis seems to fall under the category of NP-hard problem. The actual NP-hardness proof of this problem is beyond the scope of this thesis. However, the exponential time taken by MILP formulation gives an impression that the proposed problem seems to fall under the category of NP-hard problem. Therefore, the metaheuristic approach is used to solve such a hard problem. This thesis employs variable neighbourhood search (VNS) to solve the problem.

In this chapter, we describe the framework of variable neighbourhood search. The variable neighbourhood search includes shaking procedure, local search and improvement procedure.

Variable neighbourhood search is firstly proposed by Mladenović & Hansen (1997). Hansen et al. (2010) mentioned that VNS is a metaheuristic used to solve the combinatorial optimization problems. Basically, VNS consists of three processes: shaking process, local search process and neighbourhood change process (Hansen et al., 2017). In shaking process, an initial solution is randomly chosen from a neighbourhood structure. Then in the local search process, the algorithm searches for a better solution within the given neighbourhood structure. If a better solution is found, it replaces the current solution and becomes the best solution. This strategy is called the first

movement (Hansen et al., 2017). It is also the strategy applied in this thesis. The process repeats until no improvement can be made within the neighbourhood. If no improvement can be made in the local search process, the algorithm goes to neighbourhood change process. This process makes the algorithm move to next neighbourhood structure and continue the search for a better solution. The algorithm repeats these processes until the stopping condition is fulfilled. The framework of VNS is shown in Figure 4.1.

In VNS, an initial solution S is firstly generated. The initial solution is improved using a local search process, and this solution is initialized as the best solution S^b . Secondly, in the perturbation process, values of main decision variables $X_{d,p}\ and\ \theta_{s,p}$ are changed slightly to get the perturbed solution S', and they are improved using the local search process to get the new solution S''. If the objective function of the new solution $f(S'')$ is better than best solution $f(S^b)$, then S'' becomes the best solution S^b. The algorithm repeats this process until iteration i=i_{max} and report the final solution $S^b, X_{d,p}\ and\ \theta_{s,p}$. In this part, we define two new notations PAR and OD. Here PAR represents the set of all parameters related to the problem, and OD represents all other decision variables excluding the main decision variables.

	Pseudo code of VNS framework
	Input: PAR
	Output; S^b (Best solution)
1	$S \leftarrow$ Generate initial solution
2	$S^b \leftarrow$ Local search (S)

3 **For** $itr \leftarrow 1$ **to** itr_{max} **do**

4 $S' \leftarrow$ Perturbation (S)

5 $S'' \leftarrow$ Local search (S')

6 **If** $f(S'') > f(S^b)$ **then**

7 $S^b \leftarrow S''$

8 **End if**

19 $S \leftarrow S''$

10 **End for**

11 **Return** S^b

Fig 4.1. Pseudo code of VNS framework

4.1.1 Calculate other decision variables using function $CalculateOD(PAR, X_{d,p}^{new}, \theta_{s,p}^{new})$

In our problem, a solution is represented by mainly two decision variables, $X_{d,p}$ and $\theta_{s,p}$. Other decision variables are dependent on these two variables. Once the value of two decision variables is fixed, the unique value of the other decision variable can be calculated. In this thesis, we have two main decision variables $X_{d,p}$ and $\theta_{s,p}$. Besides main decision variables, we also have other decision variables $\Omega_{s,p}, \Omega_{s,p}^{+}, \Omega_{s,p}^{-}, \theta_{s,p}^{+}, \theta_{s,p}^{-}, v_{d,p}^{add}, v_{d,p}^{del}, v_{d,p}^{chg}, \Pi_{c,q}^{+}, \Pi_{c,q}^{-}, Y_{c,q,j}$ which are derived from the main decision variables $X_{d,p}$ and $\theta_{s,p}$. In this chapter, we name these decision variables as OD(other decision variables), and the calculation process of other decision variables is calculated using the function $CalculateOD$. All other decision variables are computed according to related constraints in this process.

After $X_{d,p}$ and $\theta_{s,p}$ are generated, the other decision variables can also be generated using different constraints specified in previous chapter. The value of $\Omega_{s,p}^{-}$ can be calculated according to constraint (16) based on the value of $\theta_{s,p}$ and PAR. To generate $\Omega_{s,p}^{-}$, we calculate the summation of existing patients on the waiting list and arriving patients. Then we calculate the number of $\Omega_{s,p}^{-}$ by using the summation and $\theta_{s,p}$ so that we can know how many patients are overdue at the end of each time period. In a similar way, we can also get the number of overdue patients ($\Omega_{s,p}$) according to constraint (18) at the beginning of each time period. After we generate the value of $\Omega_{s,p}^{-}$, we can calculate the number of on-time patients who are on the waiting list, which is $\Omega_{s,p}^{+}$ by using constraint (17). When we finish generating $\Omega_{s,p}^{-}, \Omega_{s,p}^{+}$ and $\Omega_{s,p}$, the number of $\theta_{s,p}^{-}, \theta_{s,p}^{+}$ can be generated easily according to constraints (19) and (20). The number of overdue patients at the end of time period p minus overdue patients at the beginning of that period are the number of patients who become overdue in the period $\theta_{s,p}^{-}$. The number of patients treated at each period $\theta_{s,p}$ minus the treated overdue patients $\theta_{s,p}^{-}$ is the number of on-time treated patients $\theta_{s,p}^{+}$. After $\theta_{s,p}^{+}$ and $\theta_{s,p}^{-}$ are generated, we can calculate the number of $\Pi_{c,q}^{+}$ and $\Pi_{c,q}^{-}$ using constraints (25) and (26). The $\Pi_{c,q}^{+}$ is the number of time patients of costed categories in each quarter of the time horizon and similarly $\Pi_{c,q}^{-}$ is the number of overdue patients. These decision variables can be calculated using only the result of $\theta_{s,p}^{+}$ and $\theta_{s,p}^{-}$ easily. By using this function, the calculation of OD can be represented clearly in the VNS algorithm.

4.1.2 Solution representation using function $\boldsymbol{SOL\ REP}(\boldsymbol{X_{d,p}^{new}}, \boldsymbol{\theta_{s,p}^{new}}, \boldsymbol{OD^{new}})$

In this subsection, we introduce a function called $SOL\ REP$ to represent a solution. The solution of operating theatre scheduling problem consists of main decision variables $X_{d,p}, \theta_{s,p}$ and other decision variables OD. After the value of one of the main decision variables changes, the solution changes with it. We use function $SOL\ REP(X_{d,p}^{new}, \theta_{s,p}^{new}, OD^{new})$ to describe this process. This function represents that a solution S is basically a collection of all decision variables.

4.1.3 Modified objective function

In this subsection, the modified objective function is proposed. Our algorithm deals with the infeasible solution. In the proposed VNS, an infeasible neighbour solution is generated by the perturbation scheme. Because of the violation caused by constraints in the problem, the perturb solution could be infeasible. The local search then tries to improve this infeasible solution and at the same time the local search try to bring the infeasible solution towards feasible solution. Therefore, we add the violation penalty in the objective function to move the infeasible solution towards the feasible solution. To solve the infeasibility, modified objective function $\bar{f}(s)$ is proposed. The violation penalty is calculated in $\bar{f}(s)$ and the formula is shown as below.

$$\bar{f}(s) = f(s) + \Delta_{TotalSession} * \sigma_{TotalSession} + \Delta_{AddedSession} * \sigma_{AddedSession}$$
$$+ \Delta_{DeletedSession} * \sigma_{DeletedSession} + \Delta_{changedSession} * \sigma_{ChangedSession}$$
$$+ \Delta_{SugeryTime} * \sigma_{SurgeryTime} + \Delta_{Ward} * \sigma_{ward} + \Delta_{ICU} * \sigma_{ICU}$$
$$+ \Delta_{MaxSession} * \sigma_{MaxSession} + \Delta_{OT} * \sigma_{OT} + \Delta_{Surgeon} * \sigma_{Surgeon}$$
$$+ \Delta_{Surgerytype1} * \sigma_{SurgeryType1} \quad (28)$$

In equation (28), $f(s)$ is the net-revenue (i.e., the original objective function of the problem) and Δ is the violation penalty for each violation and σ is the violation for each constraint. Violation of total sessions $\sigma_{TotalSession}$ is calculated based on the constraint (6). Added sessions, deleted sessions and changed sessions violations are calculated by using constraints (8), (9) and (10). We optimize surgery time in departments in constraints (11) and (12), so the violation of surgery time is based on them. In terms of downstream constraints, wards violation is calculated by constraint (13) and ICU rooms is calculated by constraint (14). The number of sessions allocated to the departments are limited in constraint (20), so violation of max allocated sessions is calculated by it. Calculations of operating theatre violation and surgeon violation are based on constraints (21) and (22). The last violation is about the overtime patients of surgery type 1, and it can be calculated by using constraint (23).

We set a high violation penalty value to move the solution towards a feasible solution. The violation penalty in this thesis is dynamic with the increment of iterations. At the beginning of each iteration, we check the feasibility of the solution. If the solution is infeasible, then we check the violation of each constraint and multiply the violation

penalty by 1.1. In this way, the algorithm will bind the constraint more in next iteration to make it feasible. Moreover, if the solution is feasible at the beginning of the iteration, then the violation penalty is divided by 1.5 and continue with the search process.

4.2 Initial solution

We divide generating initial solution process into two parts. In the first part, which is shown in *Algorithm Generate sessions* in Figure 4.2 a, we generate the number of sessions allocated to the department $X_{d,p}$. As mentioned in constraints (21) and (23), the value of $X_{d,p}$ can't exceed the limit of sessions allocated to the department D_d and available surgeons of each department in each period $F_{d,p}$. As a result, we compare the value of D_d and $F_{d,p}$ and take the bigger one as the V_{max}value in each department and period. A random number ranging from min value, which is 0, to V_{max} is generated for $X_{d,p}$.

Algorithm: Generate sessions

Input: PAR

$V_{min} \leftarrow 0$

For $i \leftarrow 1$ **to** nd **do**

  For $j \leftarrow 1$ **to** np **do**

    If $DD_i < FDP_{i,j}$

    $V_{max} \leftarrow DD_i$

    Else

    $V_{max} \leftarrow FDP_{i,j}$

    End if

9 $X_{i,j} \leftarrow UNIF(V_{min}, V_{max})$

10 **End for**

11 **End for** // Generate $X_{d,p}$

Output: $X_{d,p}$

Fig 4.2 a: Pseudo code of generate sessions

Algorithm: Generate number of treated patients

Input: PAR

1 **For** $p \leftarrow 1$ **to** np **do**

2 **For** $d \leftarrow 1$ **to** nd **do**

3 $TTAD \leftarrow SESSL \times X_{d,p}$

4 **For** $itr \leftarrow 1$ *to* itr_{max} **do**

5 **For** $k \leftarrow 1$ to NSD_d **do**

6 $s \leftarrow SDO_{d,k}$

7 $TTAD \leftarrow TTAD - TAU_s$

8 **If** $TTAD < 0$

9 **Break**

10 **Else**

11 $\theta_{s,p}++$

12 **End if**

13 **End For**

14 **End for**

15 **End for**

16 **End for** //Generate $\theta_{s,p}$

Output: $\theta_{s,p}$

Fig 4.2 b: Pseudo code of generate number of treated patients

Algorithm: Generate Initial Solution

Input: PAR, OD

Output: S^{ini}

1 $X_{d,p} \leftarrow$ Algorithm Generate sessions

2 $\theta_{s,p} \leftarrow$ Algorithm Generate number of treated patients

3 $OD \leftarrow CalculateODV(PAR, X_{d,p}, \theta_{s,p})$ //calculate other decision variables

$S \leftarrow X_{d,p}, \theta_{s,p}, OD$ //current solution representation

4 $S^{ini} \leftarrow$ Local Search (S)

Output: S^{ini}

Fig 4.2 c: Pseudo code of generate an initial solution

Secondly, we generate initial solutions for the number of patients receiving surgery $\theta_{s,p}$ using algorithm described in Figure 4.2 b. The process is shown in *Algorithm Generate number of treated patients*. For a given department, the number of patients from constraint (11), we can see that the sum of the duration of surgeries in a department can not exceed the total time allocated to the department in each period. As a result, we try to assign patients' surgery time within the limit of time allocated to each department. Thus, we make $TTAD\ (Total\ Time\ Allocated\ to\ Department) = SESSL \times X_{d,p}$, which is the time allocated to department d in time period p. Because in a department, the total time of surgeries performed during period p can't exceed the time allocated to the department. The main theme of this algorithm is to allocate the session time to each surgery equally by respecting the total time available at the department. Therefore, each surgery is incremented by 1 at a time in a sequential manner till the available time at

the department. In each iteration, the duration of selected surgery s is subtracted from $TTAD$ until $TTAD < 0$, which means that the duration of surgeries exceeds the limit. If $TTAD < 0$, the algorithm breaks and moves to the next department and period. Otherwise, the value of $\theta_{s,p}$ incremented by 1, and the algorithm repeats the process until $k < 0$. In this way, the initial solutions of $\theta_{s,p}$ are generated.

Figure 4.2 c, combines *Algorithm Generate sessions* and *Algorithm Generate number of treated patients*. After $X_{d,p}$ and $\theta_{s,p}$ are generated, OD can also be calculated. By using $X_{d,p}$, $\theta_{s,p}$ and OD, we can get the current solution. Then after the local search process, the initial solution S^{ini} is generated by the algorithm.

4.3 Perturbation scheme

After generating the initial solution, the algorithm goes to the perturbation process. In *perturbation scheme*, the value of two main decision variables $X_{d,p}$ and $\theta_{s,p}$ are changed slightly. At first, we decide how many decision variables are perturbated. $NumXpert$ represent the number of session length decision variables modified during the perturbation scheme. And $Num\theta pert$ represents the number of treated patients modified in the perturbation scheme. After that, we determine the lower limit and upper limit of the increment for decision variables, which are $3, 5$ for $X_{d,p}$ and 6, 15 for $\theta_{s,p}$. A value ranges from the lower limit to the upper limit is randomly chosen and assigned to certain $X_{d,p}$ and $\theta_{s,p}$. The process is shown in the pseudo code described in Figure 4.3.

Algorithm: Perturbation scheme

Input: PAR

For $k \leftarrow 1$ **to** $NumXpert$ **do**

  $d \leftarrow RAND(1, ND)$

  $p \leftarrow RAND(1, NP)$

  $X_{d,p}^{pert} \leftarrow RAND(3,5)$

End for

For $k \leftarrow 1$ **to** $Num\theta pert$ **do**

  $s \leftarrow RAND(1, NS)$

  $p \leftarrow RAND(1, NP)$

  $\theta_{s,p}^{pert} \leftarrow RAND(6,15)$

End For

$OD^{new} \leftarrow CalculateOD(PAR, X_{d,p}^{pert}, \theta_{s,p}^{pert})$

$S \leftarrow SOL\ REP(\ X_{d,p}^{pert}, \theta_{s,p}^{pert}, OD^{new})$

Output: S

Fig 4.3 Pseudo code of perturbation

When the perturbation process completes, the local search process starts. In the local search part, we have five different local searches in the algorithm.

4.4 Local search

This section will introduce the local search of our algorithm in detail. Four local searches are included in this thesis. The main theme of these local search schemes is to change the value of two main decision variables $X_{d,p}$ and $\theta_{s,p}$ for possible

improvement in the solution.

4.4.1 Session change local search

The first local search is the *Session Change Local Search*. In this local search, we change the value of $X_{d,p}$ by 1 unit for the possible improvement. The pseudo code of *Session Change Local Search* is shown in Figure 4.4.

Algorithm: Session Change Local Search

Input: PAR, S

Output: S

For $d \leftarrow 1$ **to** nd **do**
  For $p \leftarrow 1$ **to** np **do**
    $X_{d,p}^{new} \leftarrow X_{d,p} + 1$
    $OD^{new} \leftarrow CalculateOD(PAR, X_{d,p}^{new}, \theta_{s,p})$
    $S' \leftarrow SOL\ REP(\ X_{d,p}^{new}, \theta_{s,p}, OD^{new})$
    If $\bar{f}(S') > \bar{f}(S)$
      $S \leftarrow S'$
    End if
    $X_{d,p}^{new} \leftarrow X_{d,p} - 1$
    $OD^{new} \leftarrow CalculateOD(PAR, X_{d,p}^{new}, \theta_{s,p})$
    $S' \leftarrow SOL\ REP(\ X_{d,p}^{new}, \theta_{s,p}, OD^{new})$
    If $\bar{f}(S') > f(S)$
      $S \leftarrow S'$
    End if

15 **End for**

16 **End For**

Output: S

Fig 4.4. Pseudo code of Session Change Local Search

In this local search, for each department and period, we increment the session $X_{d,p}$ by 1 and get a new session $X_{d,p}^{new}$. Based on $X_{d,p}^{new}$, we can calculate other decision variables OD^{new}. After OD^{new} is given, new solution S' is updated. If the objective function of the new solution $\bar{f}(S')$ is greater than the objective function of the current solution $\bar{f}(S)$, then the new solution S' replaces the current solution S. Then we decrement the $X_{d,p}$ by 1 unit and repeat the same process. When $department$ and $period$ loop ends, the best solution is given and becomes the new best solution S^b.

4.4.2 Treated patient change local search

The second local search is *Treated Patient Change Local Search*. In this local search, we change the value of $\theta_{s,p}$ by 1 for possible improvement. The pseudo code of *Treated Patient Change Local Search* is shown in Figure 4.5.

Algorithm: Treated Patient Change Local Search

Input: PAR, S

Output: S

1 **For** $d \leftarrow 1$ **to** nd **do**

2 **For** $p \leftarrow 1$ **to** np **do**

3 $\theta_{s,p}^{new} \leftarrow \theta_{s,p} + 1$

4 $OD^{new} \leftarrow CalculateOD(PAR, X_{d,p}, \theta_{s,p}^{new})$

5 $S' \leftarrow SOL\ REP(X_{d,p}, \theta_{s,p}^{new}, OD^{new})$

6 **If** $\bar{f}(S') > \bar{f}(S)$

7 $S \leftarrow S'$

8 **End if**

9 $\theta_{s,p}^{new} \leftarrow \theta_{s,p}^{new} - 1$

11 $OD^{new} \leftarrow CalculateOD(PAR, X_{d,p}, \theta_{s,p}^{new})$

12 $S' \leftarrow SOL\ REP(\ X_{d,p}, \theta_{s,p}^{new}, OD^{new})$

13 **If** $\bar{f}(S') > f(S)$

14 $S \leftarrow S'$

15 **End if**

16 **End for**

18 **End For**

Output: S

Fig 4.5. Pseudo code of Treated Patient Change Local Search

For each surgery type and period, we increase $\theta_{s,p}$ by 1 and get $\theta_{s,p}^{new}$. After that we can calculate other decision variables OD^{new}**.** Then new solution S' is updated based on $\theta_{s,p}^{new}, X_{d,p}$ and OD^{new}. If the objective function of the new solution $\bar{f}(S')$ is greater than the objective function of the current solution $\bar{f}(S)$, the new solution S' replaces the current solution S. Then we decrease $\theta_{s,p}$ by 1 and repeat the same process. When $surgery\ type$ and the $period$ loop ends, the current solution becomes the improved

solution of this local search. There is a possibility that we could not find a better solution during the whole search process. In this case, the original solution is returned at the end of the local search process.

4.4.3 Joint treated patient and session change local search

The third local search is the *Joint Treated Patient and Session Change Local Search*. We change the value of sessions $X_{d,p}$ and treated patients $\theta_{s,p}$ by 1 unit at the same time in this local search. The process is shown in Figure 4.6.

Algorithm: Joint Treated Patient and Session Change Local Search

Input: PAR, S

Output: S

For $d \leftarrow 1$ **to** nd **do**
  For $j \leftarrow 1$ to NSD_d **do**
    $s \leftarrow SDO_{d,j}$
    For $p \leftarrow 1$ **to** np **do**
      $X_{d,p}^{new} \leftarrow X_{d,p} + 1$
      $\theta_{s,p}^{new} \leftarrow \theta_{s,p} + 1$
      $OD^{new} \leftarrow CalculateOD(PAR, X_{d,p}^{new}, \theta_{s,p}^{new})$
      $S' \leftarrow SOL\ REP(X_{d,p}^{new}, \theta_{s,p}^{new}, OD^{new})$
      If $\bar{f}(S') > \bar{f}(S)$
        $S \leftarrow S'$
      End if
      $\theta_{s,p}^{new} \leftarrow \theta_{s,p} - 1$

14 $OD^{new} \leftarrow CalculateOD(PAR, X_{d,p}^{new}, \theta_{s,p}^{new})$

15 $S' \leftarrow SOL\ REP(X_{d,p}^{new}, \theta_{s,p}^{new}, OD^{new})$

16 **If** $\bar{f}(S') > \bar{f}(S)$

17 $S \leftarrow S'$

18 **End if**

19 $X_{d,p}^{new} \leftarrow X_{d,p} - 1$

21 $\theta_{s,p}^{new} \leftarrow \theta_{s,p} + 1$

22 $OD^{new} \leftarrow CalculateOD(PAR, X_{d,p}^{new}, \theta_{s,p}^{new})$

23 $S' \leftarrow SOL\ REP(X_{d,p}^{new}, \theta_{s,p}^{new}, OD^{new})$

24 **If** $\bar{f}(S') > \bar{f}(S)$

25 $S \leftarrow S'$

26 **End if**

28 $\theta_{s,p}^{new} \leftarrow \theta_{s,p} - 1$

29 $OD^{new} \leftarrow CalculateOD(PAR, X_{d,p}^{new}, \theta_{s,p}^{new})$

30 $S' \leftarrow SOL\ REP(X_{d,p}^{new}, \theta_{s,p}^{new}, OD^{new})$

31 **If** $\bar{f}(S') > \bar{f}(S)$

32 $S \leftarrow S'$

33 **End if**

34 **End for**

35 **End for**

36 **End for**

Output: *S*

Fig 4.6. Pseudo code of Joint Treated Patient and Session Change Local Search

In this local search, we increase $X_{d,p}$ by 1 and get $X_{d,p}^{new}$ firstly. After that, we first

increase the $\theta_{s,p}$ by 1 and get $\theta_{s,p}^{new}$. In this way, we can update the value of OD^{new} to generate a new solution S'. Compared with the current objective function $\bar{f}(S)$, if the new objective function $\bar{f}(S')$ is greater, then S' becomes the current solution. After we finish the loop of $\theta_{s,p} + 1$, then we test $\theta_{s,p} - 1$ in the same process and update the current solution. Generally, we change the value of each $X_{d,p}$ and $\theta_{s,p}$ at the same time. Thus, all departments and surgery types are tested. After we increase $X_{d,p}$ by 1, the same process repeats with the decrement of $X_{d,p}$ by 1. The best solution of this local search is output when the value of all $X_{d,p}$ and $\theta_{s,p}$ are changed by 1.

4.4.4 Joint treated patients and session multiple change local search

The last local search is Joint *Treated Patients and Session Multiple Change Local Search*. We change the value of sessions $X_{d,p}$ by 1 and the value of treated patients $\theta_{s,p}$ by $\delta\theta$ at the same time in this local search. The process is shown in Figure 4.7.

Algorithm: Joint Treated Patients and Session Multiple Change Local Search

Input: PAR, S

Output: S

1 **For** $d \leftarrow 1$ **to** nd **do**

2 **For** $j \leftarrow 1$ to NSD_d **do**

3 $s \leftarrow SDO_{d,j}$

4 **For** $p \leftarrow 1$ **to** np **do**

5 $X_{d,p}^{new} \leftarrow X_{d,p} + 1$

6 $SurgeryTime_p \leftarrow \tau_s \times \theta_{s,p}$

7 $\delta\theta \leftarrow (SurgeryTime_p - SESSL * X_{d,p}^{new})/\tau_s$

8 $\theta_{s,p}^{new} \leftarrow \theta_{s,p} + \delta\theta$

9 $OD^{new} \leftarrow CalculateOD(PAR, \theta_{s,p}^{new}, X_{d,p}^{new})$

11 $S' \leftarrow SOL\ REP(\ \theta_{s,p}^{new}, X_{d,p}^{new}, OD^{new})$

12 **If** $\bar{f}(S') > \bar{f}(S)$

13 $S \leftarrow S'$

14 **End if**

15 $\theta_{s,p}^{new} \leftarrow \theta_{s,p} - \delta\theta$

16 $OD^{new} \leftarrow CalculateOD(PAR, \theta_{s,p}^{new}, X_{d,p}^{new})$

17 $S' \leftarrow SOL\ REP(\ \theta_{s,p}^{new}, X_{d,p}^{new}, OD^{new})$

18 **If** $\bar{f}(S') > \bar{f}(S)$

19 $S \leftarrow S'$

21 **End if**

22 $X_{d,p}^{new} \leftarrow X_{d,p} - 1$

23 $\theta_{s,p}^{new} \leftarrow \theta_{s,p} + \delta\theta$

24 $OD^{new} \leftarrow CalculateOD(PAR, \theta_{s,p}^{new}, X_{d,p}^{new})$

25 $S' \leftarrow SOL\ REP(\ \theta_{s,p}^{new}, X_{d,p}^{new}, OD^{new})$

26 **If** $\bar{f}(S') > \bar{f}(S)$

28 $S \leftarrow S'$

29 **End if**

30 $\theta_{s,p}^{new} \leftarrow \theta_{s,p} - \delta\theta$

31 $OD^{new} \leftarrow CalculateOD(PAR, \theta_{s,p}^{new}, X_{d,p}^{new})$

32 $S' \leftarrow SOL\ REP(\ \theta_{s,p}^{new}, X_{d,p}^{new}, OD^{new})$

33 **If** $\bar{f}(S') > \bar{f}(S)$

34 $S \leftarrow S'$

35 **End if**

36 **End for**

37 **End for**

38 **End for**

Output: S

Fig 4.7. Pseudo code of Joint Treated Patients and Session Multiple Change Local Search

The process of *Joint Treated Patients and Session Multiple Change Local Search* is same with *Joint Session and Patient Change Local Search*. The only difference between them is that we change the value of $\theta_{s,p}$ by more than 1 unit. The aim of this local search is to solve the infeasibility problem. When we design the algorithm, we find that sometimes changing the value of $\theta_{s,p}$ only by 1 keeps the solution infeasible because of the two conflicting constraints (12) and (23). Thus, the solution is trapped inside the infeasible region. Therefore, we calculate the value of $\theta_{s,p}$ as much as possible, which is defined by constraints (11) and (12). We calculate the total surgery time $SurgeryTime_p$ in each period for each department. Then, we use $(SurgeryTime_p - SESSL * X_{d,p}^{new})/\tau_S$ to calculate how many treated patients should be changed from original $\theta_{s,p}$, which is $\delta\theta$. In this way, it can help the algorithm find a feasible solution in fewer iterations.

Chapter 5

Numerical experiment

In this part, we present the numerical experiment for MILP formulation and VNS algorithm. The MILP formulation is solved using the CPLEX software. By solving the MILP formulation, we get the optimal solution for the problem. However, CPLEX software can solve only small problem instances. To better understand the behaviour of the algorithm, bigger problem instances are solved using VNS. The VNS algorithm is coded in C++ language and runs on Intel Core i7 2.8GHz with 16 GB of RAM. The CPLEX is used in an environment with i7-8700k CPU and 16GB RAM.

5.1 Dataset generation

We generate 2 sets of datasets to conduct numerical experiments for the problem. The first set includes 80 small instances, while the second set includes 60 large instances. To generate the dataset, we contacted Prof. Hamideh Anjomshoa, who is the author of the paper we refer to and obtain their original dataset. Their original dataset has 28 time periods, 11 departments, and 126 surgery types. Based on their datasets, we created our data set, which replicates statistically to the original data set. We generated data set for different combinations of np, nd and ns. By generating data set with different values of np, we can check how the algorithm works in terms of different lengths. In a realistic

scenario, hospitals may also need results for different time spans like 1 month or 3 months. The reason for generating data set for different values of nd and ns is that we want this algorithm to be flexible so that it can be adapted in hospitals of different sizes. Each hospital could have a different number of departments and perform a different number of surgery types. In this way, the scalability of the algorithm can be tested.

Some parameters are the same as the original dataset. Firstly, three urgency categories are considered. Secondly, the length of each session is 240 minutes. Thirdly, the length of stay for urgency categories 1,2,3 are 30,90 and 365 days. Besides these parameters, we recalculate other parameters according to the statistical pattern of the original data set. In the original dataset, the value of τ_s is a sequence ranging from 15 to 480, and the interval of 15. We calculate the percentage of each value and generate τ_s in our dataset according to the same percentage. For example, among 126 surgery types in the original dataset, 4 of them has $\tau_s = 15$ so we can know that the percentage is 3%. As a result, when we generate a new dataset with 100 surgery types, we randomly choose three surgery types and make τ_s equal to 15. In the same way, we calculate $ICU_s, LOS_s, F_{d,p}, R_s, a_{s,p}, e_{s,p}$.

As a result, all 60 small instance datasets and 80 large instance datasets are generated in this way.

5.2 Experiment of small instances

In small instances, we generate 80 datasets that are different from each other on the

number of departments (nd), the number of time periods (np) and the number of surgery types (ns). The ranges of nd are 5 and 10. The ranges of np are 12, 16, 20, 24 and 28. The ranges of ns are 20, 25, 30, 35 ,40, 45, 50 and 55. Thus, a total of 80 problem instances are generated for each combination of the department, time period and surgery. We collect the running time and solution from both CPLEX and our VNS algorithm. Net-revenue is measured by the amount of money (in dollars) the hospital earns. We also calculate the percentage Gap (PG) of the VNS solution from the optimal solution. The percentage gap is used to evaluate the performance of the proposed VNS algorithm. The percentage gap represents how far the solution is from the optimal solution. We calculate the percentage gap by using the formulation shown below.

$$PG = (Optimal\ Result - VNS\ Result) \times 100 \div Optimal\ Result \quad (29)$$

Instance No.	Number of Department nd	Number of Period np	Number of Surgeries ns	Optimal Solution		VNS		
				Net-Revenue (in dollars)	CPU time	Net-Revenue (in dollars)	CPU time	PG
1	5	12	20	101024	2.23	59577	153	41.03
2	5	12	25	92699	2.58	90271	157	2.62
3	5	12	30	49168	3.60	40277	167	18.08
4	5	12	35	67224	2.19	46757	223	30.45
5	5	12	40	53475	3.00	43581	212	18.50
6	5	12	45	58242	2.67	38716	255	33.53
7	5	12	50	77305	2.74	59619	336	22.88
8	5	12	55	102906	2.32	71367	346	30.65
9	10	12	20	106926	2.05	78534	142	26.55
10	10	12	25	95853	2.46	89750	157	6.37

11	10	12	30	58806	30.39	48648	181	17.27
12	10	12	35	85437	2.40	60062	223	29.70
13	10	12	40	56671	3.75	46019	211	18.80
14	10	12	45	75471	12.44	59568	253	21.07
15	10	12	50	92613	4.32	82830	309	10.56
16	10	12	55	109990	2.64	74849	327	31.95
17	5	16	20	36356	45.00	26327	141	27.59
18	5	16	25	61832	2.82	45734	174	26.04
19	5	16	30	96114	3.25	60911	249	36.63
20	5	16	35	79279	2.17	60411	252	23.80
21	5	16	40	103680	2.79	73897	300	28.73
22	5	16	45	87134	3.00	64404	408	26.09
23	5	16	50	204463	2.41	146154	928	28.52
24	5	16	55	110944	2.36	74767	532	32.61
25	10	16	20	48430	6.44	33571	158	30.68
26	10	16	25	70592	4.51	59319	176	15.97
27	10	16	30	99320	4.10	74326	235	25.17
28	10	16	35	90147	2.60	74471	256	17.39
29	10	16	40	103719	4.37	74320	314	28.34
30	10	16	45	128764	3.02	98359	374	23.61
31	10	16	50	240472	2.52	190680	476	20.71
32	10	16	55	166815	6.54	124610	493	25.30
33	5	20	20	97145	11.42	87508	197	9.92
34	5	20	25	149379	2.22	114178	290	23.56
35	5	20	30	111580	23.26	85685	287	23.21
36	5	20	35	96131	24.72	73785	299	23.25
37	5	20	40	107100	6.50	76666	350	28.42
38	5	20	45	100091	2.30	67670	481	32.39
39	5	20	50	103611	5.15	80765	460	22.05
40	5	20	55	133229	2.55	101098	561	24.12
41	10	20	20	110433	3.36	102269	188	7.39
42	10	20	25	174109	2.27	118774	253	31.78
43	10	20	30	119840	4.44	89048	291	25.69
44	10	20	35	103152	16.95	85677	318	16.94

45	10	20	40	135893	7.82	107073	347	21.21
46	10	20	45	144807	5.24	113489	413	21.63
47	10	20	50	138392	4.83	110263	469	20.33
48	10	20	55	158753	5.61	134277	525	15.42
49	5	24	20	124329	2.50	100857	245	18.88
50	5	24	25	80310	5.64	51836	266	35.46
51	5	24	30	139212	2.45	96815	336	30.45
52	5	24	35	154615	2.84	108180	475	30.03
53	5	24	40	229800	3.16	133657	474	41.84
54	5	24	45	141683	6.79	118787	470	16.16
55	5	24	50	107138	20.33	89500	532	16.46
56	5	24	55	131746	3.34	96264	543	26.93
57	10	24	20	155835	2.68	140358	244	9.93
58	10	24	25	114842	3.83	97761	254	14.87
59	10	24	30	161409	4.36	137230	373	14.98
60	10	24	35	174418	2.80	161424	416	7.45
61	10	24	40	266648	5.18	183551	457	31.16
62	10	24	45	183964	6.14	154238	497	16.16
63	10	24	50	138363	8.59	116845	545	15.55
64	10	24	55	169366	6.14	139989	520	17.35
65	5	28	20	192008	2.24	173512	370	9.63
66	5	28	25	215036	2.29	193632	424	9.95
67	5	28	30	161796	243.23	128860	387	20.36
68	5	28	35	155984	8.94	129038	424	17.27
69	5	28	40	144946	6.98	108668	458	25.03
70	5	28	45	151964	6.82	88419	612	41.82
71	5	28	50	323261	2.88	291516	1529	9.82
72	5	28	55	217600	3.55	175668	951	19.27
73	10	28	20	231000	4.19	166418	339	27.96
74	10	28	25	111263	82.00	92179	289	17.15
75	10	28	30	180385	17.85	150397	438	16.62
76	10	28	35	171063	5.55	137209	469	19.79
77	10	28	40	147460	6.54	125879	464	14.64

78	10	28	45	157419	50.75	131748	542	16.31
79	10	28	50	391849	3.42	363008	1124	7.36
80	10	28	55	248805	7.81	215186	783	13.51
Average				133388	10.63	105244	388.71	21.10

Table 5.1 Results of small instances experiment

From table 5.1, we can see that the average optimal net-revenue is \$133388, and the average net-revenue given by our VNS algorithm is \$105244. The percentage gap ranges from 2.62 to 41.82. The average percentage gap is 21%. The average percentage gap of 21 % indicates the complexity of the problem. It seems that the operating theatre scheduling problem is difficult to solve.

We also group the datasets by nd, np and ns to see the performance trends of the algorithm on the basis of the number of departments, periods and surgeries. All the results are shown in tables 5.2 5.3, and 5.4. The trend along with nd, np, ns is shown in figures 5.1, 5.2 and 5.3.

From table 5.2 and figure 5.1, we can see that when the number of departments increases, the percentage gap between the optimal solution and the VNS solution decreases. However, we only have two groups of nd; the trend still needs further research.

From table 5.3 and figure 5.2, we find that after the time period reaches 16, the percentage gap decreases along with the increment of the number of time periods. Therefore, we can assume that the VNS algorithm performs better when it is applied to the long-term operating room scheduling problem.

The last dataset is the group by the number of surgery type. According to table 5.4 and figure 5.3, we don't find a clear pattern of the trend when the surgery type increases. It seems that the number of surgery type increase does not have any impact on the performance of the VNS algorithm. The performance remains consistent with different surgery types.

Number of Department (nd)	Optimal Solution	VNS solution	Percentage Gap
5	123788	94383	23.75
10	142987	116105	18.80

Table 5.2 Result of small instances grouped by number of department (nd)

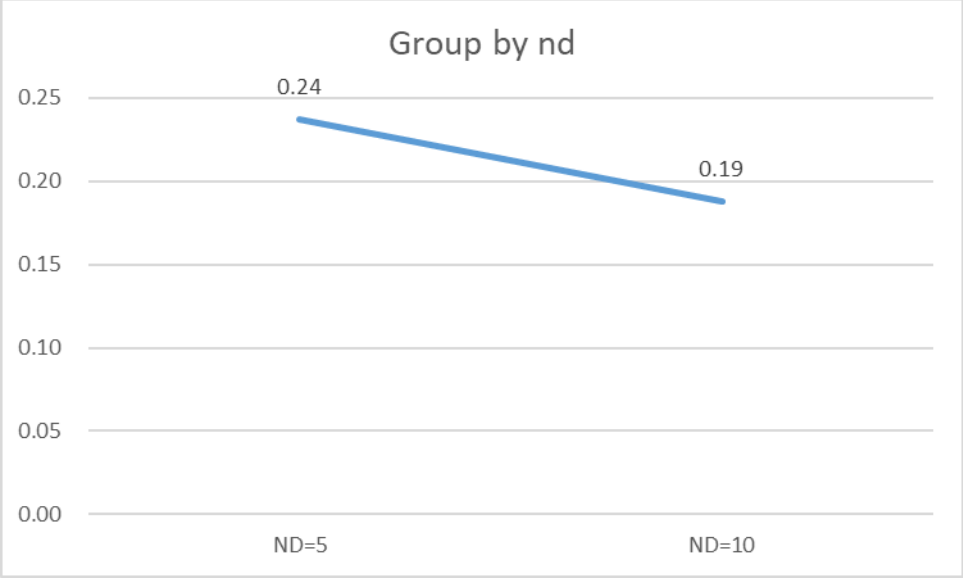

Fig 5.1 Result of small instances grouped by number of department (nd)

Number of time period (np)	Optimal Solution	VNS solution	Percentage Gap

12	80238	61901	22.85
16	108003	80141	25.80
20	123977	96764	21.95
24	154604	120455	22.09
28	200114	166958	16.57

Table 5.3 Result of small instances grouped by number of time period (np)

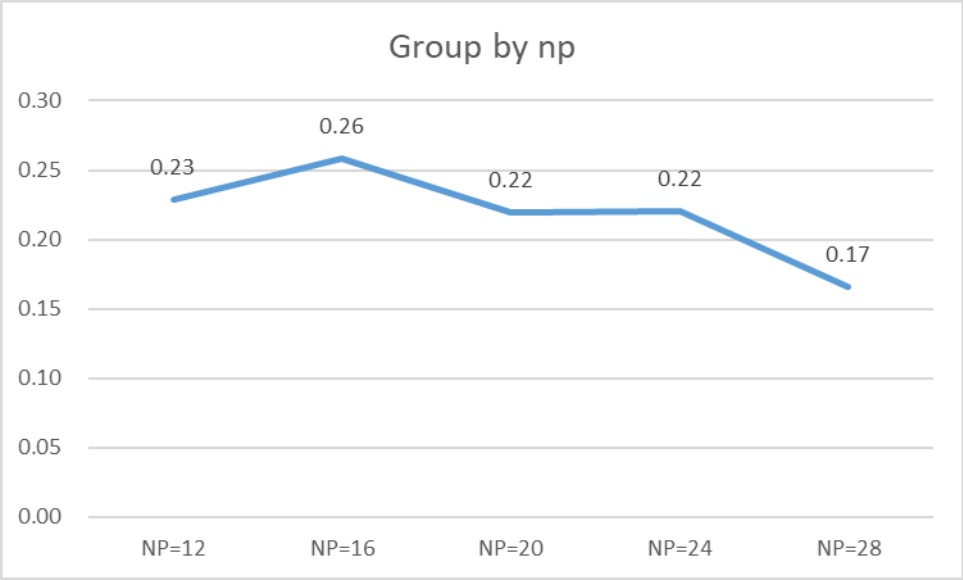

Fig 5.2 Result of small instances grouped by number of time period (np)

Number of surgery type (ns)	Optimal Solution	VNS solution	Percentage Gap
20	120348	96893	19.49
25	116591	95343	18.22
30	117763	91219	22.54
35	117745	93701	20.42
40	134939	97331	27.87
45	122953	93539	23.92
50	181746	153118	15.75
55	155015	120807	22.07

Table 5.4 Result of small instances grouped by number of surgery type (ns)

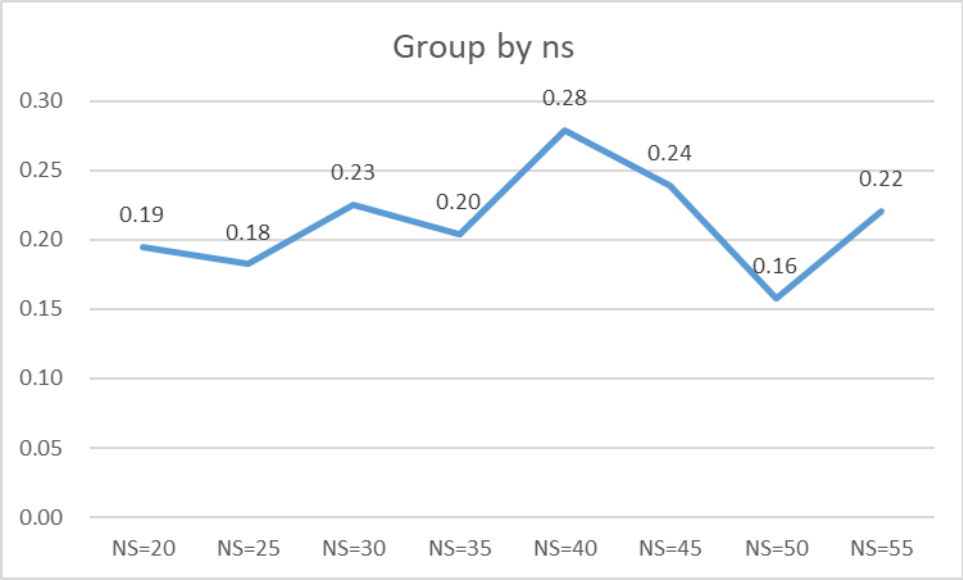

Fig 5.3 Result of small instances grouped by number of surgery type (ns)

The average running time of the optimal solution is 10.63 seconds, and the average CPU time of VNS algorithm is 388.71 seconds. Although CPLEX takes less CPU time to obtain the optimal solutions for small instances, finding the solution using CPLEX for bigger instances is difficult. In a practical scenario, hospitals can be of different sizes. Hospitals with more departments could have more surgery types, and it will make the problem more complicated. Moreover, hospitals may also want to get solutions for a longer time period, which can also increase the complexity of the problem. Under these circumstances, our VNS algorithm can still give us solutions within a reasonable time. An experiment of large instances is shown in the next subsection.

5.3 Experiment of large instances

In large instances, we generate 60 datasets that are different from each other on the number of departments (nd), number of time periods (np) and number of surgery types (ns). The ranges of nd are 5, 10 and 15. The ranges of np are 16, 28, 44, and 56. The ranges of NS are 50, 75, 100, 125 and 150. Thus, a total of 60 problem instances are generated for different combinations of departments, periods and surgeries. We compare the initial solution with the final solution of the VNS algorithm to evaluate the improvement done by the VNS algorithm. The results are shown in table 5.5. We calculate the improvement percentage by using the equation (30) shown below.

$$PG = (VNS\ solution - initial\ solution) \times 100 \div initial\ solution \quad (30)$$

Instance No	Number of Department nd	Number of Period np	Number of Surgeries ns	Initial solution	VNS solution	Improvement Percentage
1	5	16	50	80252	156772	95.35
2	10	16	50	120948	192058	58.79
3	15	16	50	98898	126388	27.80
4	5	28	50	116499	294787	153.04
5	10	28	50	211226	368117	74.28
6	15	28	50	246849	294883	19.46
7	5	44	50	172836	264743	53.18
8	10	44	50	235601	314927	33.67
9	15	44	50	285472	375154	31.42
10	5	56	50	170618	261531	53.28
11	10	56	50	230077	323525	40.62
12	15	56	50	326493	383777	17.55
13	5	16	75	41293	96979	134.86
14	10	16	75	72709	138375	90.31
15	15	16	75	104636	148972	42.37
16	5	28	75	83622	180370	115.70

17	10	28	75	134035	223549	66.78
18	15	28	75	213312	281524	31.98
19	5	44	75	171179	263967	54.21
20	10	44	75	234488	304359	29.80
21	15	44	75	306463	375586	22.56
22	5	56	75	214108	552600	158.09
23	10	56	75	335479	671024	100.02
24	15	56	75	548277	847812	54.63
25	5	16	100	52035	131062	151.87
26	10	16	100	104710	166028	58.56
27	15	16	100	155297	226552	45.88
28	5	28	100	61541	128572	108.92
29	10	28	100	114360	166281	45.40
30	15	28	100	197593	235296	19.08
31	5	44	100	215473	357887	66.09
32	10	44	100	279608	388952	39.11
33	15	44	100	335875	468202	39.40
34	5	56	100	200016	473827	136.89
35	10	56	100	298001	650010	118.12
36	15	56	100	416951	712440	70.87
37	5	16	125	54345	110632	103.57
38	10	16	125	89042	161296	81.15
39	15	16	125	128479	186393	45.08
40	5	28	125	72011	136986	90.23
41	10	28	125	133643	183391	37.22
42	15	28	125	169779	223731	31.78
43	5	44	125	201738	501663	148.67
44	10	44	125	397131	679908	71.20
45	15	44	125	439313	705404	60.57
46	5	56	125	268019	567527	111.75
47	10	56	125	393133	744334	89.33
48	15	56	125	527711	713012	35.11
49	5	16	150	56457	146310	159.15
50	10	16	150	94100	157942	67.84
51	15	16	150	125187	184243	47.17
52	5	28	150	76531	163706	113.91
53	10	28	150	166442	238624	43.37
54	15	28	150	208981	276449	32.28
55	5	44	150	153455	290302	89.18
56	10	44	150	304002	412270	35.61
57	15	44	150	363163	466427	28.43
58	5	56	150	190318	349123	83.44
59	10	56	150	280842	483768	72.26
60	15	56	150	431905	533153	23.44
Average				208542	336058.	61.15

Table 5.5 Results of small instances experiment

From the table 5.5, we can see that compared with the initial solution, the solution of the VNS algorithm increases by 61.15%. The improvement percentage ranges from 17.54% to 159.15%. The purpose of this experiment is to check how our VNS algorithm works on large problem instances. From the results, we can see that compared with initial solutions, solutions of VNS algorithm improve significantly, which means that the VNS algorithm is effective when it's applied to the large problem instances. In a practical scenario, we can expect the good performance of our VNS algorithm. The initial solution can be considered as a heuristic solution. The results indicate that the VNS is able to improve the heuristic solution by on average 61.15 %, which proves the effectiveness of the proposed VNS algorithm.

We also perform analysis to see the trend of algorithms performance on the basis of the number of departments, the number of periods and the number of surgeries. The analysis is performed in tables 5.6, 5.7, 5.8 and figures 5.4, 5.5, 5.6.

From table 5.6 and figure 5.4, we can see that with the increasing number of departments, percentage improvement decreases significantly, which means that the VNS algorithm performs better on improving the initial solution when it is applied to problems with fewer departments.

According to table 5.7 and figure 5.5, the percentage of improvement decreases when the number of time periods increases from 16 to 44. However, when the number of time

periods increases to 56, the percentage of improvement increases from 50.64 to 71.10.

The trend of results with increment of surgery type is shown in table 5.8 and figure 5.6. We find that when the number of surgery type ranges from 75 to 125, our VNS algorithm improves the solution by 66% to 71%. When the number of surgery types is less than 75 or more than 125, the percentage of improvement becomes lower. As a result, when the number of surgery type ranges from 75 to 125, the improvement of VNS algorithm from the initial solution becomes higher.

Number of Department (nd)	Initial Solution	VNS solution	Improvement Percentage
5	132617	271467	104.70
10	211478	348436	64.76
15	281531	388269	37.91

Table 5.6 Result of large instances grouped by number of department (nd)

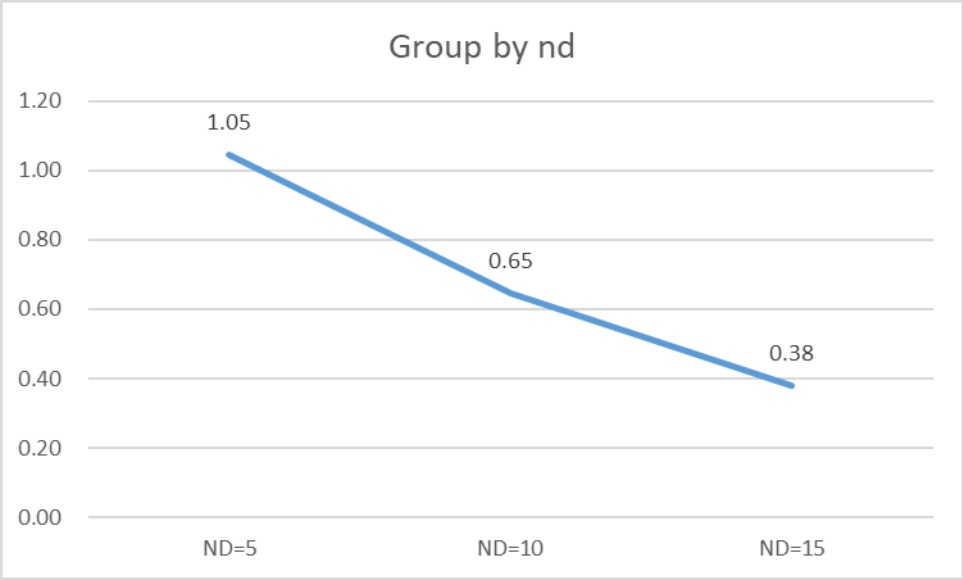

Fig 5.4 Result of large instances grouped by number of department (nd)

Number of time period (np)	Initial Solution	VNS solution	Improvement Percentage
16	91892	155333	69.04
28	147094	226417	53.93
44	273053	411316	50.64
56	322129	551164	71.10

Table 5.7 Result of large instances grouped by number of time period (np)

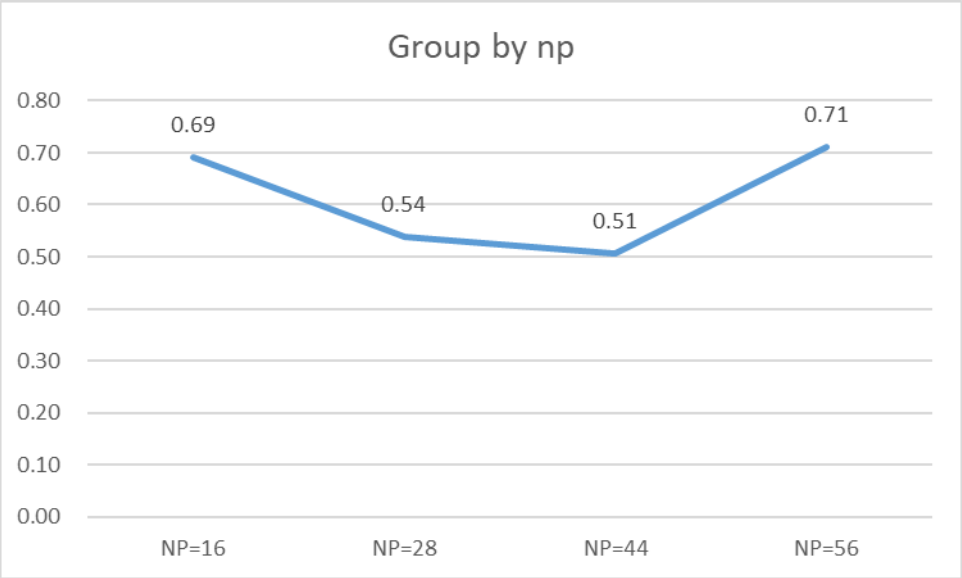

Fig 5.5 Result of large instances grouped by number of time period (np)

Number of surgery type (ns)	Initial Solution	VNS solution	Improvement Percentage
50	191314	279721	46.21
75	204966	340426	66.09
100	202621	342092	68.83

125	239528	409523	70.97
150	204281	308526	51.03

Table 5.8 Result of large instances grouped by number of surgery type (ns)

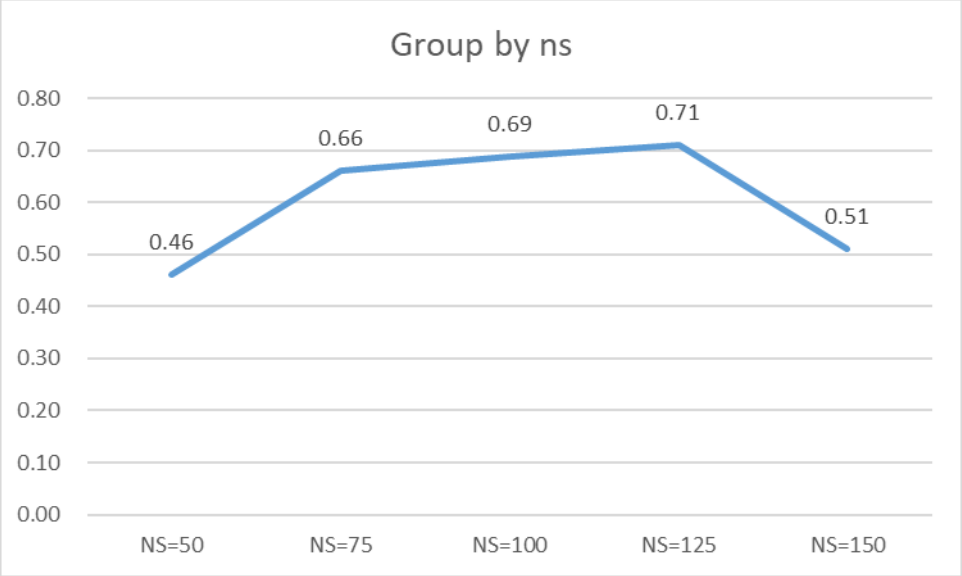

Fig 5.6 Result of large instances group by number of surgery type (ns)

5.4 Sensitivity experiment to understand problem structure

This section presents a sensitivity analysis to better understand the problem structure. A hospital management would like to know the extra benefit brought by relaxing some of the resource availability.

The first relaxed constraint is the number of ICU rooms ($NICU$). By relaxing $NICU$, we want to know the relation between the number of ICU rooms and the overall net-revenue of the hospital. This analysis will help hospitals to determine the best number of ICU rooms. This analysis will also help to make a decision on the affordability of

creating new ICU rooms or renting new ICU rooms from the third-party service provider.

The second relaxed constraint is the number of available surgeons. After conducting a sensitivity experiment, we can know how much it can help to improve the net-revenue of the hospital. As a result, hospitals can decide if they should hire more full-time and part-time surgeons.

Besides relaxing constraints, we also conduct a sensitivity experiment on the tradeoff between revenue maximization Vs penalty minimization. Revenue is excluded in the experiment to find out the revenue loss if the hospital only concentrates on treating more patients and improving performance to minimize the penalty.

5.4.1 Experiment on number of ICU rooms ($NICU$)

In this part, we consider the situation that a hospital has more ICU rooms than the existing number of ICU rooms. Hospitals with plenty of ICU rooms are also considered as one of the scenarios. For example, a hospital that only performs surgeries of low-risk surgery type can regard the number of ICU rooms as an unimportant restriction so that the constraint can be relaxed from the model. In such a situation, it can be assumed that the opening of ICU rooms for low-risks surgery is minimum and the hospital can afford to open as many new ICU rooms as needed. We increase the number of ICU rooms by 2, 5, 10, 15, 20 and report the results, respectively. Because we want to maximize the net-revenue, here we consider the weight in our objective function as the penalty of overdue patients, tardy days of overdue patients and patients on the waiting list. In this

way, we could know how much it will cost to improve the overall performance of the hospital. By performing this experiment, the hospital manager could know how many ICU rooms should be built in the hospital to balance the profit and performance. We calculated the improvement percentage from the base solution of the VNS algorithm as well. Besides net-revenue, we also calculate revenue (objective1), reward (objective2) and penalty (including objectives (3) (4) and (5)). Improvement of each component is calculated accordingly. All results are shown in table 5.9. A line chart is also drawn to make the pattern of improvement percentage clearer in figure 5.7.

	Net Revenue (in dollars)	Revenue	Reward	Penalty	Improvement Percentage from the base model			
					Net-Revenue (in dollars)	Revenue	Reward	Penalty
Base model	328992	330498.4	23.36	4182.83	0	0	0	0
2 more ICU	344291	348616.1	23.58	4348.56	5.11	5.48	0.93	3.96
5 more ICU	365539	369349.8	23.65	3833.53	11.90	11.76	1.21	-8.35
10 more ICU	391028	395004.2	23.80	3999.73	20.12	19.52	1.85	-4.38
15 more ICU	409677	414193.1	23.76	4542.30	26.23	25.32	1.68	8.59
20 more ICU	423853	428651.5	23.78	4855.25	30.63	29.70	1.77	16.08
∞ ICU	450479	455626.5	23.78	5170.57	38.11	37.86	1.77	23.61

Table 5.9 Analysis on number of ICU rooms

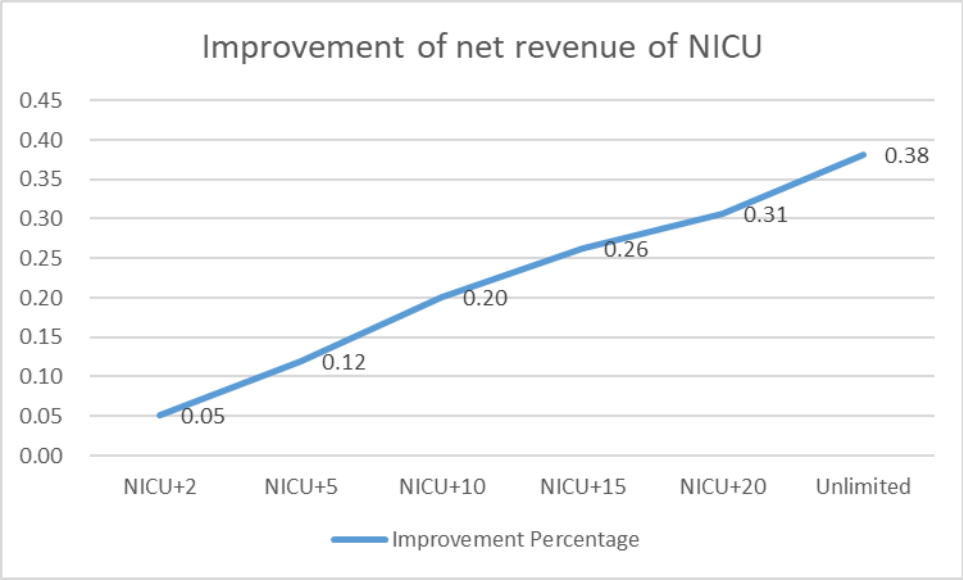

Fig 5.7 Analysis on number of ICU rooms

From the results, we can see that with the increasing number of $NICU$ by 2, 5, 10, 15, 20 and infinite, the net-revenue keeps increasing up to 38.11 %. For example, by building 20 more ICU rooms, the net-revenue improves by 30.63%. The results indicate that a hospital is able to spend 30.63 % of their current revue to build 20 more ICU rooms. Since the cost depends on the location, hence our analysis can't provide an exact recommendation on opening 20 more ICU rooms. However, our analysis gives an idea that if the cost of opening 20 more ICU rooms is within 30.63 % of the current revenue, then the hospital can go ahead with opening 20 more ICU rooms. We also find that with the increase of ICU rooms, revenue increases from 348616.1 to 455626.5 as well.

The improvement of reward improves slightly and ranges from 0.93% to 1.85%. The improvement percentage improves along with the increase of $NICU$ generally. However, the penalty decreases when the number of ICU rooms increases from 2 to 5 and increases along with the number of ICU rooms after that. According to this result,

we can know that when the number of ICU rooms increases, revenue increases along with it, but the penalty doesn't increase all the time. This result is surprising for us since having more ICU rooms should decrease the penalty. But it decreases only up to some extent, and after that, it does not decrease. One of the reasons is that after some point of ICU room relaxation, other constraints start binding the solution and hence stop the benefit brought by relaxing ICU room constraint. We should consider the pattern of penalty more in detail when we plan to build more ICU rooms.

These results could help managers to determine the number of ICU rooms so that both hospitals and patients can get benefit from it. If the number of ICU rooms is not a constraint, then the result improves by 38.11%. As a result, in our problem, the number of ICU rooms influences the results significantly. For extant hospitals, adding more ICU rooms could be an effective way to improve the overall performance. For new hospitals, building more ICU rooms could be an idea that needs to be considered carefully.

5.4.2 Experiment on the number of available surgeons.

This section performs an experiment on surgeons' availability constraints. The number of surgeons could be dynamic in hospitals. For example, when flu season comes, hospitals may need more doctors than the normal season so that the rising cases can be treated. Sometimes it's not possible to increase the number of doctors due to their limited availability. However, their working hours can be increased, which represents an increase in the number of doctors for our analysis. Hospitals can also hire part-time

surgeons so that the number of available surgeons is large enough to deal with all the patients. On the other hand, by changing this constraint, we can also know if a hospital should hire more full-time or part-time surgeons. In this section, we consider the weight of overdue patients, tardy days of overdue patients and patients on the waiting list as penalties in objective function. We could know how much hiring more surgeons influences net-revenue by performing the experiment. In table 5.10, we remove the constraint of available surgeons from the algorithm and compare the result with the original solution. The improvement percentage from the base solution is also calculated and shown in table 5.10. Table 5.11 also presents the average value of net-revenue, revenue, rewards and penalty, and how much it improves from the base model.

	Base Model	Infinite Surgeon	Improvement Percentage%
1	146154	147307	0.78
2	190680	196779	3.19
3	120439	122450	1.66
4	291516	295680	1.42
5	363008	365087	0.57
6	291237	384907	32.16
7	261864	259669	-0.83
8	303005	308746	1.89
9	371620	373385	0.47
10	249152	246571	-1.03
11	316341	318974	0.83
12	373132	371945	-0.31
13	90738	92930	2.41

14	137569	133797	-2.74
15	148972	145408	-2.39
16	176470	176955	0.27
17	220512	222052	0.69
18	277904	275574	-0.83
19	251981	255750	1.49
20	299794	298949	-0.28
21	365035	366718	0.46
22	552600	561071	1.53
23	645404	662090	2.58
24	837951	841922	0.47
25	128634	133229	3.57
26	164371	164143	-0.13
27	226552	224068	-1.09
28	125996	126412	0.33
29	165858	167826	1.18
30	230169	232272	0.91
31	345474	354076	2.48
32	388952	391649	0.69
33	443056	448160	1.15
34	449662	444617	-1.12
35	650010	649645	-0.05
36	707344	708797	0.20
37	108721	107784	-0.86
38	158171	158384	0.13
39	182468	181815	-0.35
40	134862	135241	0.28
41	181703	181982	0.15
42	222956	223977	0.45
43	492784	500794	1.62

44	679908	683392	0.51
45	690691	701656	1.58
46	554143	577056	4.13
47	696196	737763	5.97
48	701383	716468	2.15
49	142821	143058	0.16
50	157805	156652	-0.73
51	177199	179651	1.38
52	157559	158875	0.83
53	232560	235688	1.34
54	268362	267580	-0.29
55	287866	279964	-2.74
56	407599	406220	-0.33
57	466427	472778	1.36
58	335342	338188	0.84
59	470097	466601	-0.74
60	522787	520763	-0.38
	328993	333366	1.32

Table 5.10 Analysis on number of available surgeons

	Base Model (in dollars)	Infinite Surgeon (in dollars)	Improvement Percentage%
Net-Revenue	328993	333366	1.32
Revenue	330498.4	337890.2	2.23
Reward	23.36	23.41	0.21
Penalty	4182.83	4548.56	8.74

Table 5.11 Average result of an experiment on a number of available surgeons

The results reported in Table 5.11 show that hiring more surgeons in the hospital affects the net-revenue by only 1.32% compared with the base solution. We can also see that when revenue improves by 2.23%, reward only improves by 0.21%. However, the penalty improves by 8.74%, according to table 5.11. That means when we remove the limitation of the surgeon, the improvement percentage of penalty is greater than revenue and reward. One of the reasons is that when surgeon constraint is removed, more surgeons are assigned to perform surgeries which can bring more revenue.

In the scenario of our problem, the number of available surgeons doesn't influence the solution notably. Improvement percentage ranges from -2.7% to 32.16%, which means under some circumstances, removing surgeon constraints can reduce net-revenue slightly. However, we can also see that in some scenarios this constraint can be critical to the result like dataset 6. As a result, we could use VNS as a method to evaluate if a hospital needs to hire more surgeons. Hospital managers can depend on it to determine when is the good time to hire more surgeons. However, the overall results give an impression that the number of surgeon is not a bottleneck resource in the operating room theatre problem. There is also a chance that the problem instances considered in this thesis do not impose a bottleneck on the number of surgeons. We replicate the problem instances considered by Anjomshoa et al. (2018), and there is a chance that the original problem has sufficient surgeons to carry out the operations. The analysis needs further experiment to gain more insight into the impact of the number of surgeon in the

net-revenue with new data sets.

5.4.3 Analysis of revenue maximization versus penalty minimization

Besides experimenting on constraints, we also conduct the experiment on revenue maximization Vs penalty minimization. The purpose of this analysis is to assess net-revenue loss if the hospital focuses on minimizing the penalty instead of maximizing its revenue.

In order to perform this analysis, revenue is removed from the model in this part. However, the final solution includes revenue as well for a fair comparison purpose. In a realistic scenario, hospitals can be faced with serious conditions like pandemic. Under these conditions, hospitals have to overlook revenue and only focus on treating as many patients as possible and improving the performance. Charity based hospitals don't regard increasing revenue as an important performance. The purpose of charity hospitals is to treat more patients on time and provide patients with better service. As a result, it is necessary to conduct the experiment on revenue to see how much it costs to concentrate on patient satisfaction only.

We remove revenue from the objective functions to assess how much revenue loss occurs for a hospital if they only concentrate on treating patients with minimum penalty. Net-revenue for minimizing penalty is the result without consideration of revenue in the algorithm. Net-revenue for maximization is the result of considering revenue as an objective function. However, in both scenarios, the net revue is calculated at the end for the fair comparison purpose. In table 5.12, we show the value of revenue when we

exclude it in the model and include it in the model. Reward includes objective functions (1) and (2), and penalty includes objective functions (3), (4) and (5). In table 5.13, the average value of net-revenue, revenue, reward and penalty is shown.

$$Revenue\ lost = (Result\ for\ minimizing\ penalty - Net\ revenue\ for\ maximization) * 100 / Net\ revenue\ for\ maximization \quad (30)$$

	Net-revenue for minimizing the penalty (in dollars)	Net-revenue for maximizing net-revenue (in dollars)	Revenue lost Percentage
1	72487	148581	-51.21
2	114292	191364	-40.27
3	80015	122673	-34.77
4	104787	303592	-65.48
5	168740	366779	-53.99
6	182072	292806	-37.81
7	132034	267314	-50.60
8	153410	328222	-53.26
9	242106	374663	-35.38
10	166549	258297	-35.52
11	190464	326234	-41.61
12	254325	396327	-35.82
13	53511	96023	-44.27
14	85818	141632	-39.40
15	97494	150734	-35.32
16	90933	178425	-49.03
17	110737	222309	-50.18

18	153942	279073	-44.83
19	142828	253516	-43.66
20	175675	301652	-41.76
21	261097	366589	-28.77
22	213810	556721	-61.59
23	298293	647009	-53.89
24	458665	839350	-45.35
25	66045	129851	-49.13
26	68875	166283	-58.57
27	126257	227765	-44.56
28	74709	135357	-44.80
29	112711	172194	-34.54
30	109234	233859	-53.29
31	131065	347603	-62.29
32	174531	392902	-55.57
33	202954	447100	-54.60
34	164456	451667	-63.58
35	200037	653349	-69.38
36	254034	708584	-64.14
37	49725	114264	-56.48
38	99764	162781	-38.71
39	97429	183291	-46.84
40	75762	137476	-44.89
41	104777	183447	-42.88
42	112110	224943	-50.16
43	149809	503913	-70.27
44	192369	694334	-72.29
45	242054	696361	-65.24
46	129453	555091	-76.67
47	168004	697244	-75.90

48	299091	702934	-57.45
49	61383	149455	-58.92
50	77793	159233	-51.14
51	87098	178172	-51.11
52	77611	158715	-51.10
53	139135	233699	-40.46
54	132435	269593	-50.87
55	151417	129851	16.60
56	134741	413476	-67.41
57	241463	472172	-48.86
58	163339	336895	-51.51
59	214676	471965	-54.51
60	264365	524198	-49.56
			-49.75

Table 5.12. Analysis of revenue maximization versus penalty maximization

	Net-revenue for maximization	Net-revenue for minimizing the penalty	Reduction percentage
Net Revenue (in dollars)	328992	151938	-53.81
Revenue	330498.4	152579.9	-53.83
Reward	23.36	23.1	-1.11
Penalty	4182.83	664.28	-84.11

Table 5.13. Average results of revenue maximization versus penalty maximization on average

From table 5.13, we can see that compared with the original solution, removing revenue from the algorithm affects the value of revenue significantly. When we exclude revenue, which means that we don't maximize it, the average revenue lost is 53.83%, and the average net-revenue lost is 53.81%. As a result, in our problem, if a hospital completely overlooks revenue, then they lose 53.83% of their revenue on average. Moreover, we can see that compared with net-revenue maximization, the penalty reduces by 84.11%. According to the results, although a hospital will lose 53.83% of their revenue, the penalty will also reduce by 84.11% when they focus on penalty minimization. The results can help hospitals perform a tradeoff between revenue maximization and penalty minimization. By performing this analysis, managers of hospitals could estimate how much money they will lose in emergency situations such as a pandemic or natural disaster. On the other hand, the government can also learn about how much money it will cost if they plan to build a new charity-based hospital.

Chapter 6

Conclusion

In recent years, hospitals are facing more pressure because of the larger aging population and pandemics. Operating theatre scheduling has become a problem waiting to be solved urgently. Variable neighbourhood search has been used to solve different optimization problems for a long time, but few researchers apply it to operating theatre scheduling. This thesis aims to use VNS and solve operating theatre problems effectively. Considering operating theatre scheduling involves many different resources, revenue and costs. This thesis refers to multiple objective models proposed by Anjomshoa et al. (2018) and proposes a VNS algorithm to solve the problem. From the results, we find that the percentage gap between the optimal solution and the VNS solution is 22% when it is applied to small instances problem. However, when we solve large problem instances, we find that CPLEX can't solve the problems but VNS algorithm can still give solutions in a reasonable time. As a result, our VNS algorithm can be used in big hospitals with more departments and surgery types. It can also solve problems that have a longer time period. One of the aims of this thesis is to fill the gap by applying a metaheuristic approach to solve the operating theatre scheduling problem. According to our research, our VNS algorithm performs better when it is used to solve large problem instances. Therefore, further research can focus on reducing the percentage gap, running time and improving the performance on small problem

instances. By conducting sensitivity analysis, we find that the number of ICU rooms influences the net-revenue significantly. When we add 20 more ICU rooms to the base model, net-revenue increases by 30.63%. Hospital managers can determine how many ICU rooms should be built by conducting this analysis. Moreover, we also find that in our problem, the number of available surgeons influences net-revenue very slightly in most situations, and it seems that the number of the surgeon is not a bottleneck resource for the instances considered in this thesis. But on certain occasions, available surgeons could have significant influence. By conducting the analysis, hospital managers could know when they should hire more surgeons. At last, we make an analysis on revenue maximization versus penalty minimization. According to this analysis, we could know a hospital will lose 53.83% of the revenue on average when it has to overlook revenue completely in urgent conditions like a pandemic. Practically, our VNS algorithm provides big hospitals with a new tool to prepare surgical schedule plan for operating theatre . With the demand of operations increasing, we believe that the importance of research on VNS algorithms in the operating scheduling area will improve as well.

For future research, we expect more experiment on small instance problems in order to minimize the percentage gap from optimal solution and minimize the running time of VNS algorithm. Moreover, we also think that future research can focus on elective surgery scheduling problem on tactical level with multiple objective function. Considering multiple objective functions respectively may help government supervise the service quality and performance of hospitals.

Reference

[1] Adan, & Vissers, J. M. . (2002). Patient mix optimisation in hospital admission planning: a case study. International Journal of Operations & Production Management, 22(4), 445–461. https://doi.org/10.1108/01443570210420430

[2] Ali, A., Gajpal, Y., & El Mekkawy, T. Y. (2017). Heuristics for Parallel Flowshop Scheduling Problem. ICRBS-2017, 2017.

[3] Ali, Gajpal, Y., & Elmekkawy, T. Y. (2020). Distributed permutation flowshop scheduling problem with total completion time objective. Opsearch, 58(2), 425–447. https://doi.org/10.1007/s12597-020-00484-3

[4] Anjomshoa, Dumitrescu, I., Lustig, I., & Smith, O. J. (2018). An exact approach for tactical planning and patient selection for elective surgeries. European Journal of Operational Research, 268(2),728–739. https://doi.org/10.1016/j.ejor.2018.01.048

[5] Aringhieri, R., Landa, P., Soriano, P., Tànfani, E., & Testi, A. (2015). A two level metaheuristic for the operating room scheduling and assignment problem. Computers & Operations Research, 54, 21–34. https://doi.org/10.1016/j.cor.2014.08.014

[6] Beliën, J., Demeulemeester, E., & Cardoen, B. (2009). A decision support system for cyclic master surgery scheduling with multiple objectives. Journal of Scheduling, 12(2), 147–161. https://doi.org/10.1007/s10951-008-0086-4

[7] Bhardwaj, Gajpal, Y., Surti, C., & Gill, S. S. (2020). HEART: Unrelated parallel machines problem with precedence constraints for task scheduling in cloud computing using heuristic and meta-heuristic algorithms. Software, Practice & Experience, 50(12), 2231–2251. https://doi.org/10.1002/spe.2890

[8] Bowers, & Mould, G. (2005). Ambulatory Care and Orthopaedic Capacity Planning. Health Care Management Science, 8(1),41–47. https://doi.org/10.1007/s10729-005-5215-4

[9] Castro, P. M., & Marques, I. (2015). Operating room scheduling with Generalized Disjunctive Programming. Computers & Operations Research, 64,

262–273. https://doi.org/10.1016/j.cor.2015.06.002

[10] Ceschia, S., & Schaerf, A. (2016). Dynamic patient admission scheduling with operating room constraints, flexible horizons, and patient delays. Journal of Scheduling, 19(4), 377–389. https://doi.org/10.1007/s10951-014-0407-8

[11] Chaabane, Meskens, N., Guinet, A., & Laurent, M. (2008). Comparison of two methods of operating theatre planning: Application in Belgian Hospital. Journal of Systems Science and Systems Engineering, 17(2), 171–186. https://doi.org/10.1007/s11518-008-5074-x

[12] Choi, & Wilhelm, W. E. (2014). On capacity allocation for operating rooms. Computers & Operations Research, 44, 174–184. https://doi.org/10.1016/j.cor.2013.11.007

[13] Creemers, S., Beliën, J., & Lambrecht, M. (2012). The optimal allocation of server time slots over different classes of patients. European Journal of Operational Research, 219(3), 508–521. https://doi.org/10.1016/j.ejor.2011.10.045

[14] Cardoen, B., Demeulemeester, E., & Beliën, J. (2010). Operating room planning and scheduling: A literature review. European Journal of Operational Research, 201(3), 921–932. https://doi.org/10.1016/j.ejor.2009.04.011

[15] Dellaert, N., & Jeunet, J. (2017). A variable neighbourhood search algorithm for the surgery tactical planning problem. Computers & Operations Research, 84, 216–225. https://doi.org/10.1016/j.cor.2016.05.013

[16] Fügener, Hans, E. W., Kolisch, R., Kortbeek, N., & Vanberkel, P. T. (2014). Master surgery scheduling with consideration of multiple downstream units. European Journal of Operational Research, 239(1), 227–236. https://doi.org/10.1016/j.ejor.2014.05.009

[17] Gajpal, & Rajendran, C. (2006). An ant-colony optimization algorithm for minimizing the completion-time variance of jobs in flowshops. International Journal of Production Economics, 101(2), 259–272. https://doi.org/10.1016/j.ijpe.2005.01.003

[18] Gajpal, Rajendran, C., & Ziegler, H. (2006). An ant colony algorithm for scheduling in flowshops with sequence-dependent setup times of jobs. International Journal of Advanced Manufacturing Technology, 30(5), 416–424. https://doi.org/10.1007/s00170-005-0093-y

[19] Gajpal, Y., Dua, A., & Sahu, S. N. (2014). Heuristics for single machine scheduling under competition to minimize total weighted completion time and makespan objectives. Lecture Notes in Management Science, 6, 99-105.

[20] Guido, R., & Conforti, D. (2017). A hybrid genetic approach for solving an integrated multi-objective operating room planning and scheduling problem. Computers & Operations Research, 87, 270–282. https://doi.org/10.1016/j.cor.2016.11.009

[21] Hansen, Mladenović, N., & Moreno Pérez, J. A. (2009). Variable neighbourhood search: methods and applications. Annals of Operations Research, 175(1), 367–407. https://doi.org/10.1007/s10479-009-0657-6

[22] Hansen, P., Mladenović, N., Todosijević, R., & Hanafi, S. (2017). Variable neighbourhood search: basics and variants. EURO Journal on Computational Optimization, 5(3), 423–454. https://doi.org/10.1007/s13675-016-0075-x

[23] Hulshof, P. J. H., Kortbeek, N., Boucherie, R. J., Hans, E. W., & Bakker, P. J. M. (2012). Taxonomic classification of planning decisions in health care: a structured review of the state of the art in OR/MS. Health Systems, 1(2), 129–175. https://doi.org/10.1057/hs.2012.18

[24] Islam, & Gajpal, Y. (2021). Optimization of Conventional and Green Vehicles Composition under Carbon Emission Cap. Sustainability (Basel, Switzerland), 13(12), 6940–. https://doi.org/10.3390/su13126940

[25] Islam, Gajpal, Y., & ElMekkawy, T. Y. (2021a). Mixed fleet based green clustered logistics problem under carbon emission cap. Sustainable Cities and Society, 72, 103074–. https://doi.org/10.1016/j.scs.2021.103074

[26] Islam, Gajpal, Y., & ElMekkawy, T. Y. (2021b). Hybrid particle swarm optimization algorithm for solving the clustered vehicle routing

problem. Applied Soft Computing, 110, 107655–. https://doi.org/10.1016/j.asoc.2021.107655

[27] Jebali, A., & Diabat, A. (2015). A stochastic model for operating room planning under capacity constraints. International Journal of Production Research, 53(24), 7252–7270. https://doi.org/10.1080/00207543.2015.1033500

[28] Lagergren, M. (1998). What is the role and contribution of models to management and research in the health services? A view from Europe. European Journal of Operational Research, 105(2), 257–266. https://doi.org/10.1016/S0377-2217(97)00233-6

[29] Lamiri, M., Xie, X., Dolgui, A., & Grimaud, F. (2008). A stochastic model for operating room planning with elective and emergency demand for surgery. European Journal of Operational Research, 185(3), 1026–1037. https://doi.org/10.1016/j.ejor.2006.02.057

[30] Lebowitz, P. (2003). Schedule the Short Procedure First to Improve OR Efficiency. AORN Journal, 78(4), 651,657–654,659. https://doi.org/10.1016/S0001-2092(06)60671-6

[31] Lee, S., & Yih, Y. (2014). Reducing patient-flow delays in surgical suites through determining start-times of surgical cases. European Journal of Operational Research, 238(2), 620–629. https://doi.org/10.1016/j.ejor.2014.03.043

[32] Lehtonen, J.-M., Torkki, P., Peltokorpi, A., & Moilanen, T. (2013). Increasing operating room productivity by duration categories and a newsvendor model. International Journal of Health Care Quality Assurance, 26(2), 80–92. https://doi.org/10.1108/09526861311297307

[33] Li, Gajpal, Y., & Bector, C. . (2018). Single machine scheduling with two-agent for total weighted completion time objectives. Applied Soft Computing, 70, 147–156. https://doi.org/10.1016/j.asoc.2018.05.027

[34] Li, H., Gajpal, Y., & Bector, C. R. (2020a). A survey of due-date related single-machine with two-agent scheduling problem. Journal of Industrial &

Management Optimization, 16(3), 1329.

[35] Li, Gajpal, Y., Surti, C., Cai, D., & Bhardwaj, A. K. (2020b). Nature-Inspired Metaheuristics for Two-Agent Scheduling with Due Date and Release Time. Complexity (New York, N.Y.), 2020. https://doi.org/10.1155/2020/1385049

[36] Li, Gajpal, Y., Bhardwaj, A. K., Chen, H., & Liu, Y. (2021a). Two-Agent Single Machine Order Acceptance Scheduling Problem to Maximize Net Revenue. Complexity (New York, N.Y.), 2021, 1–14. https://doi.org/10.1155/2021/6627081

[37] Li, Gajpal, Y., & Appadoo, S. S. (2021b). Algorithms for a two-agent single machine scheduling problem to minimize weighted number of tardy jobs. Journal of Information & Optimization Sciences, 42(4), 785–811. https://doi.org/10.1080/02522667.2020.1816637

[38] Macario, A., Vitez, T. S., Dunn, B., & McDonald, T. (1995). Where are the costs in perioperative care?: Analysis of hospital costs and charges for inpatient surgical care. The Journal of the American Society of Anesthesiologists, 83(6), 1138-1144. https://doi.org/10.1097/00000542-199512000-00002

[39] Mladenović, N., & Hansen, P. (1997). Variable neighbourhood search. Computers & Operations Research, 24(11), 1097–1100. https://doi.org/10.1016/S0305-0548(97)00031-2

[40] Peng, Zhang, Y., Gajpal, Y., & Chen, X. (2019). A Memetic Algorithm for the Green Vehicle Routing Problem. Sustainability (Basel, Switzerland), 11(21), 6055–. https://doi.org/10.3390/su11216055

[41] Penn, M., Potts, C., & Harper, P. (2017). Multiple criteria mixed-integer programming for incorporating multiple factors into the development of master operating theatre timetables. European Journal of Operational Research, 262(1), 194–206. https://doi.org/10.1016/j.ejor.2017.03.065

[42] Pham, D.-N., & Klinkert, A. (2008). Surgical case scheduling as a generalized job shop scheduling problem. European Journal of Operational Research, 185(3), 1011–1025. https://doi.org/10.1016/j.ejor.2006.03.059

[43] Sahu, Gajpal, Y., & Debbarma, S. (2017). Two-agent-based single-machine scheduling with switchover time to minimize total weighted completion time and makespan objectives. Annals of Operations Research, 269(1-2), 623–640. https://doi.org/10.1007/s10479-017-2515-2

[44] Testi, A., Tanfani, E., & Torre, G. (2007). A three-phase approach for operating theatre schedules. Health Care Management Science, 10(2), 163–172. https://doi.org/10.1007/s10729-007-9011-1

[45] Van der Lans, M., Hans, E. W., Hurink, J. L., Wullink, G., van Houdenhoven, M., & Kazemier, G. (2006). Anticipating urgent surgery in operating room departments. University of Twente, Tech. Rep. WP-158.

[46] Zhang, Gajpal, Y., & Appadoo, S. S. (2018). A meta-heuristic for capacitated green vehicle routing problem. Annals of Operations Research, 269(1-2), 753–771. https://doi.org/10.1007/s10479-017-2567-3

[47] Zhang, Gajpal, Y., Appadoo, S. S., & Wei, Q. (2020). Multi-Depot Green Vehicle Routing Problem to Minimize Carbon Emissions. Sustainability (Basel, Switzerland), 12(8), 3500–. https://doi.org/10.3390/su12083500

[48] Zhu, S., Fan, W., Yang, S., Pei, J., & Pardalos, P. M. (2019). Operating room planning and surgical case scheduling: a review of literature. Journal of Combinatorial Optimization, 37(3), 757–805. https://doi.org/10.1007/s10878-018-0322-6