A Simulation Based Approximate Dynamic Programming Approach to Multi-class, Multi-resource Surgical Scheduling

By

Davood Astaraky

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfilment of the requirements for the degree of
Master of Science in Systems Science

uOttawa

The University of Ottawa

Abstract

The thesis focuses on a model that seeks to address patient scheduling step of the surgical scheduling process to determine the number of surgeries to perform in a given day. Specifically, provided a master schedule that provides a cyclic breakdown of total OR availability into specific daily allocations to each surgical specialty, we look to provide a scheduling policy for all surgeries that minimizes a combination of the lead time between patient request and surgery date, overtime in the ORs and congestion in the wards. We cast the problem of generating optimal control strategies into the framework of Markov Decision Process (MDP). The Approximate Dynamic Programming (ADP) approach has been employed to solving the model which would otherwise be intractable due to the size of the state space. We assess performance of resulting policy and quality of the driven policy through simulation and we provide our policy insights and conclusions.

Acknowledgements

It is an honour for me to thank those who made this thesis possible and words fail to appreciate them in a way that it should be.

First and foremost, I would like to give my sincere gratitude to Dr. Jonathan Patrick for his extraordinary supervision, guidance, and contribution from the very early stage of this research as well as giving me valuable knowledge and experiences throughout the work. Above all and the most needed, his insightful direction, encouragement and support in various ways will never be forgotten.

My special thanks go to The Ottawa Hospital (TOH) for providing a necessary data and for verifying the validity of the model and their continuous consultation on the details of the modeling process.

I also would like truly appreciate Mr. Gilbert Arbez who shared a great deal of his experiences with me and provided me with valuable inputs and constructive during the implementation part of the work.

`

Contents

`

`

`

List of Tables

`

List of Figures

`

`

Chapter 1: Introduction

1.1 Thesis Motivation and Contributions

The challenge of providing timely access to health care services is growing not only in Canada but in almost in every country in the world (Carter 2002). Globally, public health systems face increasing and lengthy wait times for a wide range of medical services. Wait lists are the biggest political issue in Canadian healthcare(Santibanez, Begen et al. 2007). In many cases, these lengthy wait times can potentially impact the health of the patient(Sanmartin, Gendron et al. 2004). For this reason, healthcare managers and policy makers face considerable political and community pressure to better manage healthcare resources and thus reduce the wait times for medical services to acceptable levels. Increasingly within Canada, both provincial and federal directives have placed pressure on hospitals to meet pre-specified wait time targets for various procedures. In response, hospitals are forced to consider increasing available resources (capacity), limiting demand or determining ways to improve efficiency use of existing resources (Patrick 2006). In most cases, increasing capacity or limiting demand may not be possible leaving improved efficiency as the primary option.

`

Nowhere is this more important than in the surgical department due to the key role operating room (OR) scheduling plays in determining hospital occupancy levels and the fact that it is the hospital's largest cost and revenue center as well as the most resource-intensive department (Macario, Vitez et al. 1995). In practice, scheduling surgeries in a medical facility is a complex and important process, and the choice of schedule directly impacts the overall performance of the system extending well beyond the operating room (Santibanez, Begen et al. 2007). Hospitals are hampered by the fact that actual demand is often not entirely known by the hospital as it handled in each physician's office. Thus, they are often forced to base scheduling decisions on prior workloads rather than actual demand. Moreover, the actual scheduling into surgical blocks is left to the discretion of the physicians within each specialty reducing the ability of the hospital to allocate downstream resources efficiently. Though the hospital does make sure that not too many patients are booked into a block, they only take into account the length of surgery and not the length of post-operative stay.

One means of improving healthcare operations is intelligent advanced patient scheduling and capacity planning (Patrick, L. Puterman et al. 2008). Advanced patient scheduling refers to methodologies for scheduling patient appointments in advance of the service date, when future demand is still unknown.

In terms of effective surgical scheduling, while there has been some improvement, all would agree that wait times remain a concern. Figure 1 illustrates the distribution of wait times for surgery over the period of one year in our dataset.

`

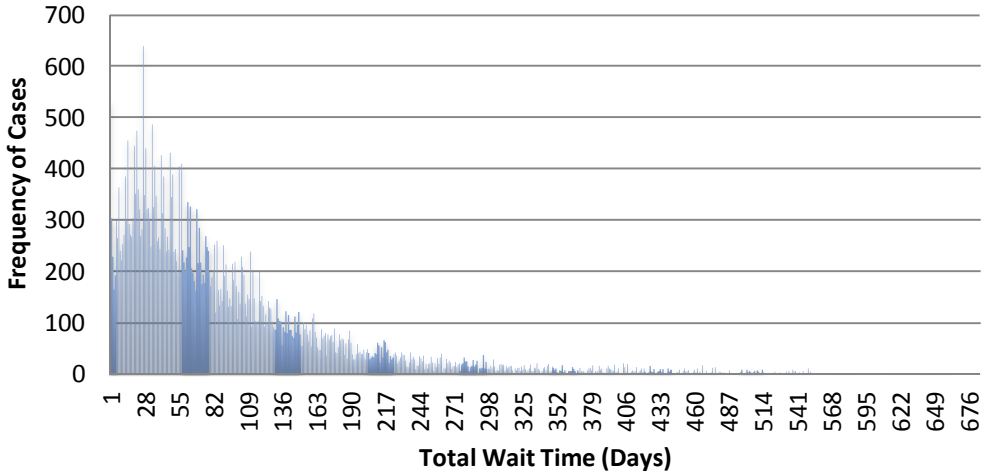


Figure 1. Distribution of Total Wait Time for Surgery

However, determining the appropriate scheduling policy and the necessary resource capacity in order to meet the wait time targets is far from straightforward. One of the major complicating factors is that each surgical patient consumes more than one type of resource and does so in a non-deterministic fashion. Before surgery, it is unknown how long the surgery will take (i.e., how much OR time will be consumed) or how long the patient will need to remain in the hospital (i.e., how many bed-days will be consumed). The arrival rate of demand and the distribution of that demand into priority classes are also stochastic. For instance, Figure 2 illustrates variability in demand arrivals over the period of one year in our dataset. In this graph X axis represent the arrival rate and Y axis represent the probability density function.

`

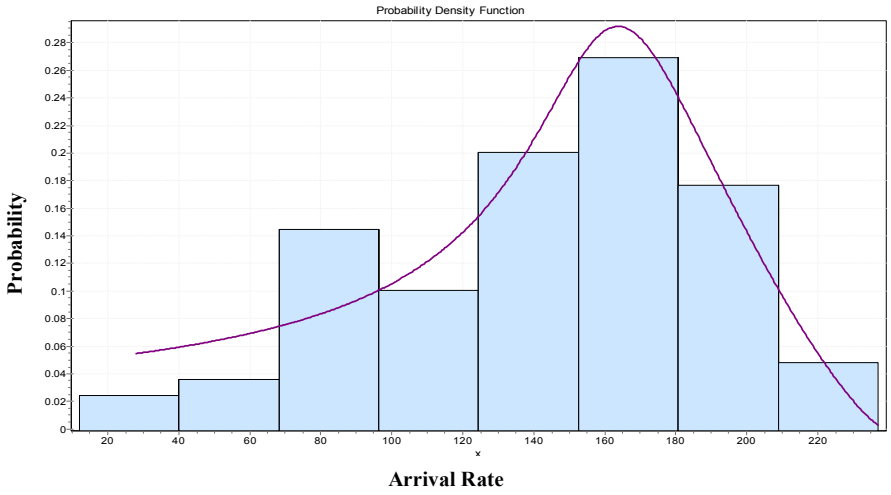


Figure 2. Variability in Demand Arrivals

In such an environment, it is difficult to determine the appropriate balance that does not allocate too much capacity (resulting in high levels of idle-time but low patient waiting times) nor too little capacity (resulting in higher patient waiting times and overtime use of resources).

Much of the work on surgical scheduling has focused primarily on the within day issues of how much time to allocate to each surgery in order to minimize physician idle time, patient wait time, and overtime on the day of surgery. It is generally agreed that a within day policy that schedules from least variable to most variable is the most advantageous for meeting these objectives although other factors may mean that such a policy must nevertheless be modified. For instance, surgeries with highly variable lengths are often the more complicated ones and it may not be advisable to leave these to the end of the day.

`

In contrast, there is a very little available research that seeks to optimize the day-to-day scheduling so that OR time and bed capacity are used efficiently. The available research that does look at this problem develop models at the specialty level (i.e., OR block and not the surgery) in order to minimize the maximum bed utilization while assuming deterministic (e.g., average) length of stay and surgery durations, and treating the problem as a single-period decision problem. Most of the studies also failed to consider any priority classification.

1.2 Surgical Scheduling Process and Problem Statement

In this section we state the research problem in more detail. In the surgical department each day a random number of requests for surgery arrive from different surgical procedures and from different patient classes. In a hospital, the patient classes are inpatients, daycare and emergency patients with different priorities and also different types of surgery. Each patient priority class has a unique maximum recommended wait time. Our focus in this research is the scheduling of elective cases as in our dataset there is a specific operating room reserved for emergency cases. In our dataset, there are nearly 900 different types of surgical procedures. The procedures types differ with respect to their surgery length and their length of stay in the hospital (LOS).

Many hospitals schedule their OR suites using cyclic manner or block schedules. In a block scheduling system, blocks of OR time are reserved for surgical specialties in a repeatable fashion. Surgeons book their cases into the allotted block time only if surgery

`

can be completed within the assigned time. The amount of block time normally is determined based on the specific surgical specialty's historical record of OR usage and based on the average of surgery length. The booking horizon is the span of time, e.g., 4 weeks (5 days each) in our case, within which requests for surgery can be booked.

The challenge facing the scheduler is how to allocate available appointment times between the different priority classes so as to minimize the number of patients whose wait time exceeds the maximum recommended wait time target. This requires significant foresight as each day's decision clearly impacts on the number of appointment slots available for future demand. If lower-urgency patients are booked too soon, then there may be insufficient capacity for later arriving higher urgency demand. Conversely, if lower-urgency patients are booked too far into the future, there would be the potential for idle capacity. The other difficult aspect is the fact that the scheduler is trying to take into account both OR time and bed utilization for each case. A poor schedule can lead to highly variable levels of bed occupancy downstream whereas a good schedule will smooth out downstream bed utilization.

To capture the trade-offs inherent in scheduling surgical patients we associated a penalty cost called a delay cost if the patient is not booked and remains in the waiting list for another day. It is intuitively obvious that a delay cost should be higher for a higher priority patient as it should be more costly to delay the booking of a higher priority patient. The use of overtime as a result of overbooking patients also leads to an additional cost. Here there is a trade-off between overbooking patient versus delaying the patient

`

that needs to be managed. Overbooking patients results in less patient waiting time but can lead to regularly exceeding the allocated OR time within the block. Conversely delaying patients results in more patient waiting time but possibly less overtime. There is also a cost associated for booking a patient after the recommended wait time target has been exceeded.

After surgery, patients stay in the recovery unit for post-operative assessment for a few hours. Hereafter the patients, day care surgical patients excepted, are transferred to the ward for post-operative care before they are discharged. How long patients remain in the ward varies according to their type of surgery and condition. Hence, scheduling and selection of patients also ought to depend on the bed availability in the ward in addition to operating room availability.

1.3 Research Objectives

Elective surgery scheduling based on a block scheduling system can be divided into four stages. In the first stage the amount of available OR time is assigned among various surgical specialties. In the next stage, the master block schedule is determined which specifies a cyclical schedule that allocates operating room block time to each major surgical specialty over the duration of the cycle. In the third stage, the number of patients that should be scheduled into each block in the master schedule is determined in order to meet the maximum recommended wait time targets that have been pre-set for each patient priority classes. At the fourth level, the sequence of surgeries to be performed and

`

the planned start times (appointment times) of each case are determined. Assuming a known amount of OR time for each specialty in stage 1, ideally one should consider all remaining three levels of decisions simultaneously. However, the complexity of treating all three together has led researchers to study them separately.

The research goal is to determine a scheduling policy for elective surgeries that seeks to meet pre-specified wait time targets while maximizing the use of available resources while minimizing associated costs. Our focus here is on the third stage of the scheduling process. The policy will schedule individual patients into each available block; a portion of the scheduling generally left to the discretion of the physicians. We are assuming the existence of a master schedule and we are not dealing with the appointment scheduling problem at the fourth stage. These elements are regarded as extensions of the project.

The rational for seeking to build a complete scheduling policy is that it is impossible to achieve efficient resource utilization with only partial control of the schedule. Concurrent with building the scheduling policy, we will look to determine optimal capacity both for OR time and for beds. In this research we focus on ward beds and ignore intensive care unit (ICU) and post-anesthesia care unit (PACU).

Setting the optimal OR capacity is a balancing act between costs associated with maintaining a certain amount of OR time, and the cost associated with utilizing overtime, and/or failing to meet the wait time targets. Setting the optimal bed capacity is similarly a balancing act between maintaining a certain number of open beds versus using overflow

`

beds. It would be best to manage these trade-offs based on real cost figures for the various resources.

The resulting models can be transported (with modified parameter values) to other hospitals and clinics as necessary. The resulting scheduling policy also can be easily integrated in a decision support tool.

The Ottawa Hospital (TOH) has helped validate the model by providing the necessary data to generate a realistic test scenario. The data include surgical patients admitted to hospital during the year 2010.

1.4 Thesis Methodology

Healthcare problems often involve high degrees of uncertainty and sequential decision making. They involve uncertainty due to the randomness of patient arrivals, resource availability and treatment duration. The common theme is that the decision maker is looking for a set of decisions called a decision policy that determines what action to take for any given scenario.

All scheduling problems are classic sequential decision problems where the decision made today impacts on what options are available tomorrow. As such it is readily modeled as a Markov Decision Process (MDP) model. MDP is a strong mathematical framework to analyze sequential decision-making problems under uncertainty. An MDP model is a system in which decisions are made sequentially overtime and future actions and outcomes depend on current and past decisions (Puterman 1994). Solving an MDP

`

provides an optimal policy as to how to allocate available appointment slots to incoming demand. In our setting capacity is described by the amount of OR time available for each specialty on each future day and demand is the number of patients from each class waiting to be booked.

In a surgical department of a hospital, each time a surgery is booked a certain amount of operating room (OR) time and a certain number of bed days are consumed. Before the surgery is performed the amount of OR time consumed is not known with certainty, and before discharging the patient from the hospital, the number of bed days consumed is not known with any certainty. Thus, we have to make booking decisions before knowing the exact resource requirements. In addition, today's decision as to when to book each waiting surgery has an inevitable impact on the available actions in future days and the cost that will be incurred. Hence, the surgical scheduling problem is a classic model of stochastic, sequential decision making and the need for a booking decision policy recommends MDP as a solution method.

Unfortunately, to determine the optimal policy for a real-sized problem, MDP models often become challenging to solve due to the computational challenges involved. To overcome such computational challenges, a new branch of operation research called *Approximate Dynamic Programing* (ADP) has evolved, developed by researchers in operations research, computer science and engineering (Bertsekas and Tsitsiklis 1996); (Sutton and Barto 1998); (Powell 2007). Simulation-based ADP is the overarching methodology used in our research for solving the underlying MDP model. ADP methods

`

provide good but not necessarily optimal policies. Hence policies obtained by ADP should be tested through simulation. We use a simulation model to determine performance indicators of the derived policy which can be compared with current practice.

1.5 Thesis Organization

The thesis progress plan is illustrated in Figure 3. The thesis is organized as follows:

In Chapter 2 we review relevant literature on "surgical scheduling and capacity planning", and discuss the similarities and differences of our research compared to the existing literature. Relevant literature regarding the Markov decision Process and Approximate Dynamic Programming methodology applied in the thesis will be reviewed in separate chapters.

Chapter 3 focuses on modeling the surgical scheduling problem as a Markov Decision Process model and the challenges and resulting solutions involved in solving the model. In section 3.1 and 3.2 we give a basic overview of the concept of a general class discounted MDPs and some history of the development of MDP models. In section 3.3, we present the comprehensive model for our problem that includes the derivation of the master schedule. Section 3.4 presents a simplified version of the surgical scheduling model that focuses on the within-block scheduling.

In Chapter 4 we present the clustering approach used to address the challenge of creating a workable number of patient classes.

`

In Chapter 5, the solution methodology which is a special type of simulation based-approximate dynamic programming is presented. Section 5.1 provides a brief introduction to the classical policy iteration method. Section 5.2 provides details of the general ADP approaches to solving MDP models and how they can be employed to solve the proposed model. Section 5.3 reviews the approximate version of the policy iteration approach and section 5.4 presents the idea of simulation based version of the approximate policy iteration which is used in this thesis. In Section 5.5 we present an algorithm developed to implement the idea of simulation based approximate policy iteration for our model and section 5.6 talks about the step size parameter chosen in the implemented algorithm

Chapter 6 presents the results of the algorithm's performance and the derived policy. Section 6.1 presents the convergence behaviour of the proposed algorithm. Sections 6.2 and 6.3 investigate the performance of the derived policy from the ADP methodology through simulation. Lastly, Section 6.4 and 6.5 discuss the advantages and drawbacks of the applied methodology and present possible future research directions.

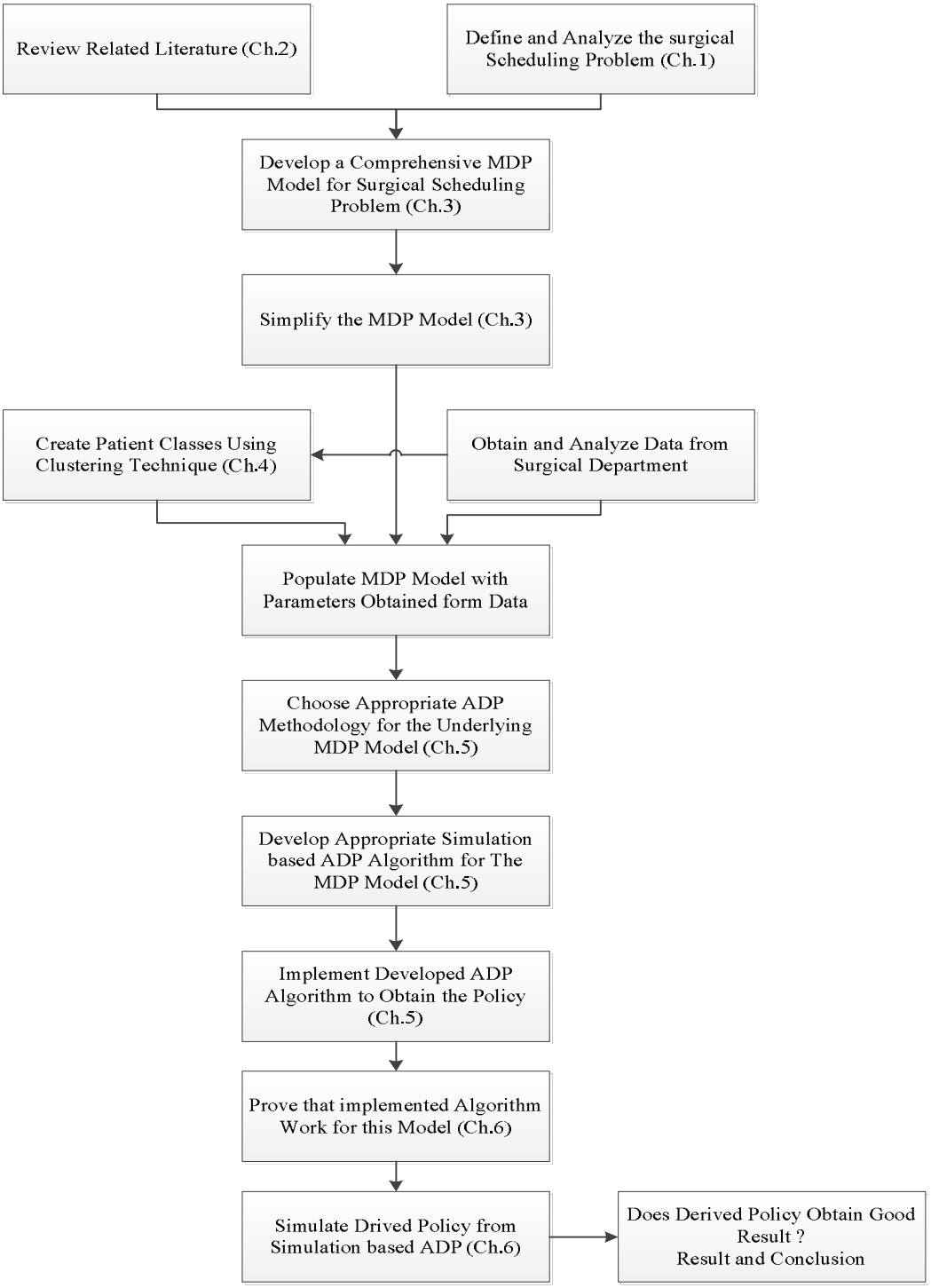


Figure 3. Thesis Progress Workflow

`

Chapter 2: Related Literature

In this chapter the relevant literature on surgical scheduling and capacity planning in health care is reviewed. We also point out to what extent related approaches differ from our own work.

2.1 Surgical Scheduling and Capacity Planning

Many hospitals schedule their OR suites using cyclic master or block schedules, in which available OR capacity is assigned to specific surgical specialties. In the context of using block schedules, the available literature on elective surgery scheduling categorizes the problem as consisting of four stages (Blake and Donald 2002), (Santibanez, Begen et al. 2007), (Testi, Tanfani et al. 2007) :

1) Determining the amount of OR time to allocate to various surgical specialties,

2) Creating a block schedule implementing the desired allocations

3) Determining the number of patients to be scheduled into each available block

4) Appointment scheduling or sequencing cases within specific ORs

The first stage is referred to as mix planning (Santibanez, Begen et al. 2007).The surgical OR block schedule is developed at the second stage. Among the available

`

literature on block surgery scheduling, (Blake and Donald 2002), (Santibanez, Begen et al. 2007), (Testi, Tanfani et al. 2007) developed the surgical block schedule which specifies which specialties will use the operating rooms each day.

Santibanez, et al. (2007) developed a mixed integer programming model to schedule surgical blocks for each specialty into ORs considering OR time availability and post-surgical resource constraints. Their objective is to present a model to explore trade-offs between OR availability, bed capacity, surgeon's booking privileges and waitlists. Their results show that without increasing post-surgical resources, the surgical department could handle more surgery cases by scheduling specialties differently. The model though is limited to consider deterministic operating times and post-operative lengths of stay and fails to include variability such as randomness in the length of stay or duration of procedures.

Zhang, et al. (2009) developed a finite-horizon mixed integer programming (MIP) model to determine a weekly operating room allocation template. They include patient priority (emergency and non-emergency) and capacity constraints in their formulation. Their objective is to minimize inpatients' cost measured as their length of stay. They test the optimal solution from the MIP model using simulation and conclude that OR utilization has improved and the average length of stay for each specialty can be reduced. The question that still remains regarding their research is that how the schedule can impact on the length of stay?

`

As mentioned in Chapter 1, the OR block schedule will serve as an input to our MDP model though we will present a version of MDP for scheduling within blocks that we do not implement that incorporates the formation of the master schedule.

All hospitals, regardless of whether they use block schedules or not, must solve the problem of scheduling individual patients to a specific OR time. This is the subject of the third and fourth stages known as patient mix where individual patients are scheduled on a daily basis that determines the number of surgeries to perform in a block, the sequence of the surgeries to be performed, and possibly their planned start time. The available literature considers these two steps together (Santibanez, Begen et al. 2007), but since in our research we focus on the third step, we consider them as separate phases and discuss each separately and call them individual patient scheduling.

The problem of optimal patient scheduling has received considerable attention in the operations research literature. The models differ in several aspects, such as the number of patient classes, the consideration of single or multi-period problems, the number of resources, the use of overbooking, the distribution of the examination length, and the objective function to be optimized. In general, individual patient scheduling studies can be categorized based on their consideration of the following three decisions: (1) choosing which surgical cases to schedule, (2) assigning cases to ORs on specific days, and (3) sequencing cases within specific ORs (Herring 2011). Typically either the first two or the last two of the above decisions are modeled. Based on which subset of the decisions is being considered, scheduling objectives range from minimizing patient waiting times, to

`

maximizing OR utilization and reducing overtime (Herring 2011). In this thesis we focus on the first two decisions but discuss the expansion of the model to incorporate the third stage.

Related papers include the work of Gerchak, et al. (1996) on allocating surgery time between elective and emergency surgeries. They address the problem of how many elective patients should be scheduled into the OR on a certain day when there is stochastic demand of emergency patients that needs to be served on the same day, and the duration of operations is random. Gerchak, et al. (1996) employ an infinite horizon MDP to solve the problem to optimality. Their objective is to maximize the expected profit that includes the revenue from the surgeries and penalty costs for using overtime and postponing a surgery to the next day. Our work differs from this paper in that Gerchak, et al. consider only two patient classes (elective and emergency) whereas we don't consider emergency cases but allow for an arbitrary number of elective patient classes. Furthermore, our work is similar in that we use the infinite horizon MDP with delay and overtime penalty in our objective, but differs in that we don't take into account profit from the surgeries. Finally, their work also does not book in advance so it is hard to understand how it can be implemented in practice.

Ozkarahan (2000) use an integer goal programming approach to select which patients to schedule and into which ORs on a single day. They develop a goal programming model which can produce schedules by minimizing idle time and overtime, and increasing satisfaction of surgeons, patients, and staff. The approach involves sorting the

`

requests for a particular day on the basis of OR utilization, block capacity, surgeon preferences and intensive care capabilities. Here the authors focus on a single day rather than the entire planning horizon. Our work differs from this paper as the focus is on how actions taken on one day affect future decisions so that we can look at the entire planning horizon rather than a single day in isolation. In contrast to models that are based on integer programming, our methodology captures the stochastic evolution of the system over time.

Ogulata and Erol (2003), Lamiri, et al. (2008), Min and Yih (2010) focus on selecting which patients are assigned to each OR over the course of a longer planning horizon, typically one week. These papers rely on a multi-stage model to address the separate decisions. Ogulata and Erol (2003) use a hierarchical multiple criteria mathematical programming approach to generate weekly operating room schedules. They breakdown the overall problem into manageable hierarchical stages: selection of patients, assignment of operations to surgeon group and determination of operation dates and operating rooms. The goals considered in these models are maximization of OR utilization, minimization of patient waiting times, and balanced distribution of operations among surgeon groups in terms of operation days. Both Lamiri, et al. (2008) and Min and Yih (2010) use two-stage stochastic programming techniques. Lamiri, et al. (2008) develops an OR planning model considering two types of demand for surgery: elective surgery and emergency surgery. Emergency cases arrive randomly and have to be performed on the day of arrival. The planning problem consists of assigning elective cases to different periods over a planning

`

horizon in order to minimize the sum of elective patient related costs and overtime costs of operating rooms. They show that the solution of this method is proved to converge to a real optimum as the computational availability increases. In the work by Min and Yih (2010), the objective is to generate an optimal surgery schedule of elective surgeries with uncertain surgery duration and the availability of downstream resources such as the surgical intensive care unit (SICU) over multi-periods. The sample average approximation (SAA) method is proposed for obtaining an optimal surgery schedule for the stochastic optimization model with respect to minimizing the total cost of patient costs and overtime costs.

In our MDP model individual patients are scheduled for surgery dynamically over time as the demand for surgery is generated resulting in a dynamic and sequential decision making process. In the context of sequential decision making processes, Patrick, et al. (2008) addresses the problem of dividing the capacity of a CT scanner between multiple outpatient classes. Their objective is to minimize the cost associated with the number of patients who fail to meet the waiting time target of their class. Our work differs in that, in the context of surgical scheduling, we are dealing with a larger number of patient classes and multiple resources with stochastic service times (ORs and downstream beds) which makes the problem more complicated.

There are a few studies that do consider the dynamic evolution of a surgical schedule with a focus on cases and waiting lists, but do so either for a small number of cases or for a limited number of days, e.g., (Dexter , Traub et al. 2003), (Dexter and Macario 2004)

`

and (Dexter and Traub 2002). To the best of our knowledge, there is no research on multi-class surgical scheduling, where the goal is to meet the mandated wait time targets while maximizing the utilization of operating room times and downstream beds.

The fourth step of the surgical scheduling process deals with the problem of *Appointment Scheduling* which determines the order and the start times for the patients previously scheduled for surgery today. An appointment schedule assigns an allocated duration by specifying the appointment time of each surgery at which the required resource and the patient will be available. However, due to the uncertain surgery durations, some surgeries may finish earlier whereas others may finish later. In the latter case, the next surgery has to wait for the preceding surgery to complete and will start later than its original appointment time (Begen 2010).

Among the available literature on sequencing cases within specific ORs, Begen and Queyranne (2011) explore the problem of Appointment Scheduling with Discrete Random Durations. They consider the problem of determining an optimal appointment schedule for a given sequence of jobs (e.g., medical procedures) on a single processor (e.g., operating room, examination facility, physician), to minimize the expected total underage and overage costs when each job has a random processing duration given by a joint discrete probability distribution. Their model can handle a given due date for total processing (e.g., end of day for an operating room) after which overtime is incurred, as well as no-shows and some emergencies.

`

Denton et al. (2007) develop a stochastic optimization model and some practical heuristics for computing OR schedules while taking into account stochastic surgery durations. They focus on the simultaneous effects of sequencing surgeries and scheduling start times. They find a simple sequencing rule based on surgery duration variance and prove that it can be used to generate substantial reductions in total surgeon and OR team waiting, OR idling, and overtime costs.

`

Chapter 3: Markov Decision Process Model for Surgical Scheduling Problem

In Chapter 1, we argued that a Markov Decision Process model is the appropriate methodology for solving the surgical scheduling problem. In this chapter we provide both the mathematical and conceptual background of a general Markov Decision Process (MDP) model .We then propose the MDP formulation for the third stage of surgical scheduling problem.

3.1 Markov Decision Process

Sequential decision-making problems are problems in which decisions are made in sequence and each decision is affected by the preceding set of decisions. A Markov Decision Process (MDP) is a mathematical framework for addressing sequential decision making problem. MDPs provide optimal solutions to this problem.

In the MDP framework, systems are characterized by a collection of variables evolving over time called the *state variables*. The state of the system contains all relevant information for making an informed decision. The *state space* is the set of all possible states of the system. To formulate a system as an MDP, the state variable must be

`

designed in a way to satisfy the important Markov property. The Markov property implies that the future state of the system depends only upon the present state and is independent of its complete past. This can be accomplished by capturing all relevant information from the past history into the definition of the current state. A decision maker can influence the state of the system by a suitable choice of the system's *actions* or *decision variables*. The decision maker observes the state of the system at specific points in time called *decision epochs* which determine how often a decision is made. Decision epochs could be at fixed intervals for discrete models or varying intervals for continuous models. The set of available actions at each state is called the *action set*. Taking a given action in a given state results in an immediate *cost* (or *reward*) which distinguishes a good action from poor one. The state of the system changes to a new state according to the set of *transition probabilities*. The transition probabilities determine the probability with which a possible state is visited at the next decision epoch based on the current state and action taken in that state. Based on the Markov property, the probability of reaching a new state is only dependent on the current state and the action taken in that state, not on the history of past states and actions.

The actions applied to the system have long term consequences. Decisions made at the current epoch have an impact on the decisions available in the next and future epochs. Therefore in making decisions, we need to balance the immediate cost with the potential of future costs. In order to solve the MDP, we need to determine which action to take at every decision epoch (called a decision rule) over a finite or infinite horizon that

`

minimizes some function of the total cost incurred over time. A sequence of these decision rules will be called a *policy*. The function of the total cost could be the total discounted cost function over the planning horizon or the long run average cost. A solution to the MDP is to find a policy that minimizes this function.

In this thesis we consider the problem of determining such decisions so as to minimize a discounted sum of costs incurred over an infinite horizon. So we consider the general class of infinite horizon, discounted MDPs running in discrete time.

We now provide a formal description of Markov decision process. A Markov decision process in discrete-time is characterized by a 4-tuple (S,U,P,C) with the following interpretation: S is the state space, U is the action space, P is the stochastic dynamics of the model where $P_{ij}(u)$ gives the transition probability from state i to state j under action u, and $C(i,u)$ gives the immediate cost of taking action u in state i. Future costs are discounted by a scalar discount factor $0<\gamma<1$. The discount factor plays an important role as a computational factor when we present the ADP framework in the next chapter. A stationary policy of the MDP is a mapping from states to actions and is denoted by $\mu: S \to U$. Given the MDP with these specifications, the goal is to minimize the value function $J^{\mu}: S \to \mathbb{R}$ over the set of admissible policies Π:

$$\underset{\mu\in\Pi}{Min} J^{\mu}(i_0) = \underset{\mu\in\Pi}{Min} E\left[\sum_{t=0}^{\infty}\gamma^t C\left(i_t,\mu\left(i_k\right)\right)\mid \mu\right] \qquad (3.1)$$

`

Here, $i_0 \in S$ is an initial state and the expectation is over all possible future states $\{i_1, i_2, ...\}$ given a fixed policy $\mu \in \Pi$. In solving the MDP, the primary interest is the policy μ^* which satisfies the minimum in equation (3.1) simultaneously for all states $i \in S$. The policy μ^* is said to be optimal policy such that:

$$J^{\mu^*}(i) \geq J^{\mu}(i) \quad \text{for all } i \in S \text{ and all } \mu \in \Pi \qquad (3.2)$$

It is proven that considering policy μ, associated value functions J^{μ} satisfies the Bellman optimality equation (Bellman 1954):

$$J^{\mu}(i) = \underset{u \in U}{Min}\left(C(i,u) + \gamma \sum_{j \in S} P_{ij}(u) J^{\mu}(j) \right) \qquad \forall i \in S \qquad (3.3)$$

Consequently, given the value function J, the associated policy μ can be found by:

$$\mu(i) = \arg \underset{u \in U}{Min}\left(C(i,u) + \gamma \sum_{j \in S} P_{ij}(u) J(j) \right) \qquad \forall i \in S \qquad (3.4)$$

It is also proven that Bellman's optimality equation has a unique solution for the value function which is the optimal value function J^*.

Finally it can be shown that the policy associated with the optimal value function J^* is an optimal policy to the MDP. Hence the optimal value function J^* associated with μ^*, satisfies the Bellman optimality equation:

$$J^*(i) = \underset{u \in U}{Min}\left(C(i,u) + \gamma \sum_{j \in S} P_{ij}(u) J^*(j) \right) \qquad \forall i \in S \qquad (3.5)$$

`

As a result, once the optimal value functions J^* are known, the optimal policy μ^* can be found by solving:

$$\mu^*(i) = \arg \underset{u \in A}{Min} \left(C(i,u) + \gamma \sum_{j \in S} P_{ij}(u) J^*(j) \right) \qquad \forall i \in S \qquad (3.6)$$

Hence in order to solve a MDP model, we need to compute J^* and determine the optimal policy associated with that. There are several methods for doing that in the dynamic programming domain. In the next chapter, we describe how an optimal policy may be found via traditional dynamic programming methods, and how the curse of dimensionality makes the application of classical dynamic programming algorithms intractable, and how approximate dynamic programming can solve the issue.

3.2 Some History

The study of Markov decision theory can be traced back to the work by Wald (1947) on sequential decision functions. In the 1940s, Bellman (1957) introduced fundamental ingredients to the Markov decision models and is credited for identifying the *Bellman optimality equations*, a central result of dynamic programming which restates an optimization problem in recursive form. Blackwell (1962) has studied discounted Markov Decision Process in great detail. Much research in the area was motivated by the book written by Howard (1960), Dynamic Programming and Markov Processes. He was the first to introduce the *policy iteration* method to find the optimal policy for MDPs. He was

`

also the first to study MDP problems with average cost. Later on, White (1963) proposed *value iteration* as a technique to find the optimal policies.

3.3 Markov Decision Process Model for Surgical Scheduling Problem

In this section we present our comprehensive MDP model for the surgical scheduling problem. In section (3.4) we propose the simplified version of the model that we have actually implemented in this thesis.

3.3.1 Decision Epochs and State Space

We assume that booking decisions are made once a day. On a given day, the scheduler will have access to the schedule as it stands from the current day to the end of the booking horizon. The scheduler bases the booking decisions on the current schedule, the current census in the hospital and the number of patients waiting to be booked. Those patients already booked are categorized according to surgical type. Let J be the number of surgical specialties. The number of surgical specialties is not the same as the number of surgical procedures. More likely, the number of surgical procedures is more detailed than the number of surgical specialties. However there is a mapping so that each patient class is contained in a single surgical specialty. We assume that there are I patient classes and that the booking horizon is N days long. Determining an appropriate set of patient classes is challenging as the number of potential classes is extremely large. We will

`

discuss the approach that is used for creating patient classes in Chapter 4. We let $I(j)$ represent the set of patient classes that are included in surgical specialty $j \in J$.

We represent the state space by a vector $\vec{s} = \{\vec{w}, \vec{x}, \vec{y}, \vec{z}\}$ that captures the booking slate, the hospital census, waiting demand and capacity allocation. We let $w_{in}, n > 0$, represent the number of patients of class i already booked for surgery in n days. We let x_{ikn} represent the number of patients of class i who are in their nth day in bed class k. Including multiple downstream bed classes allows us to differentiate beds by surgical type but more importantly to track bed capacity requirements for ICU as well as the regular surgical ward. We can always let the number of bed classes equal one if we want to ignore this added complication. We assume that no patient of class i stays more than $M(i)$ days in the hospital because intuitively the probability of a patient of class i being discharged after $M(i)$ days stabilizes to a constant value.

We let the vector, $\vec{y}$, represent waiting demand where y_i represents the number of class i patients who are waiting to be booked.

To accommodate the presence of a block schedule we also incorporate the capacity into the state space through a vector, $\vec{z}$, where $z_{jn}, n \geq 0$ represents the amount of OR capacity reserved for surgical specialty j, n days from now. If there is a previously determined fixed block schedule for the whole booking horizon we can simply remove $\vec{z}$ from the state space and assign its values as parameters for the model.

`

We assume that there is $C^B(k)$ type k beds, $k \in \{1,2,..,K\}$, assigned to surgery but that there is additional overflow capacity should it be required. Following current hospital practice, each patient of class i is assigned a surgery length, $l(i)$ that incorporates expected set up time and tear down time for that type of surgery. A patient can only be booked into a given block if adding that time to the slot does not exceed the capacity.

For ease of notation, we let $[A] = \{k,..., A\}$ and $[A] = \{1,..., A\}$.

3.3.2 Action Set

The scheduler's task is to determine which of the available slots to assign to incoming and waiting demand. The actions consist of bookings and capacity adjustments. It might be worthwhile to include cancellation of surgeries as a possible action but since this is a last resort action, we ignore it in order to build a system that hopefully does not need to use such measures.

We represent bookings by a vector, $\vec{b}$, where b_{in} represents the number of class i patients to book n days in advance. We represent capacity adjustments by the vector $\vec{c}$ where c_{jn} represents time added to or subtracted from day n for surgical specialty j. Any changes in capacity are assumed to take place by the following day. Thus, a vector of possible actions can be written as the combination of b_{in} and c_{jn} in the vector $\vec{a} = (\vec{b}, \vec{c})$. Actions are constrained by a number of factors. To be valid, any action should satisfy the following constraints:

`

First, there is a capacity constraint which insures that the capacity is not violated:

$$\sum_{i \in I(j)} l(i)\left(b_{in} + w_{in}\right) \leq z_{jn} \qquad \forall j \in [J] \qquad (3.7)$$

Second, capacity is constrained to be a multiple of a set length of time for each surgical specialty $j \in J$, $L(j)$:

$$z_{jn} + c_{jn} = kL(j) \qquad \forall (j,n) \in [J] \times [N]_0 \qquad (3.8)$$

where k is a positive integer. The rationale for this constraint is that most hospitals divide the time in one OR in a day into at most two different specialties. What that set length of time is may depend on the surgical specialty as some elective specialties may require a full day in an OR while others can make do with half a day.

Third, for each patient class there may be inadmissible days (e.g., emergency patients are not booked in advance and elective patients may require a certain number of days between request and procedure date for the pre-operative work):

$$b_{in} = 0 \qquad \forall (i,n) \in [I] \times E(i) \qquad (3.9)$$

Where $E(i)$ is the set of inadmissible booking dates for patients of class i.

Fourth, bookings need to be less than waiting demand:

$$\sum_{n \in [N]_0} b_{in} \leq y_i \qquad \forall i \in [I] \qquad (3.10)$$

Fifth, all bookings are required to be integer and positive:

$$b_{in} \in \mathbb{Z}^+ \qquad \forall (i,n) \in [I] \times [N]_0 \qquad (3.11)$$

Therefore, we can denote the action set for any given state by:

`

$$A_{\vec{S}} = \Big\{(\vec{b},\vec{c}) \mid$$

$$\sum_{i\in I(j)} l(i)\big(b_{in} + w_{in}\big) \le z_{jn}, z_{jn} + c_{jn} = kL(j), \sum_{n\in[N]_0} b_{in} \le y_i, (\vec{b},\vec{c}) \ge 0, Integer\Big\} \qquad (3.12)$$

3.3.3 Fundamental Dynamics

Stochastic elements of the process are new arrivals, discharges and transfers between bed classes. The model is complicated by the fact that the horizon is not static but rolling as day 2 becomes day 1 and so on after each decision epoch. We present the transition of each part of the state vector separately, beginning with the transition of the booking schedule:

$$\vec{w} = \{w_{i0},...,w_{iN}\} \to \Big\{w_{i1} - D^w_{i1} + b_{i1},...,w_{iN} - D^w_{iN} + b_{iN}, 0\Big\}_{i\in[I]} \qquad (3.13)$$

where D^w_{in} represents discharges from the booking schedule for patients from class i who are n days away from surgery. Since no one is booked more than N days in advance, the N^{th} day is always empty.

The hospital census transitions according to:

$$\vec{x} = \Big\{x_{ik1},...,x_{ikM(i)}\Big\}_{(i,k)\in[I]\times[K]} \to$$

$$\left\{W_{ik} + \sum_{m\in[K]} T_{ikm}, x_{ik1} - D^x_{ik1},...,x_{ik,M(i)-1} - D^x_{ik,M(i)-1} + x_{ik,M(i)} - D^x_{ik,M(i)}\right\}_{(i,k)\in[I]\times[K]} \qquad (3.14)$$

`

where W_{ik} is the proportion of w_{i0} that transit to a bed of class k post-surgery, D^x_{ikn} represents patients from class i in bed class k who are discharged after n days (either from the hospital or to another bed class) and T_{ikm} represents incoming transfers from bed class m. Here we are assuming that the LOS is unaffected by previous stays in other bed classes. This formulation allows us to potentially include PACU as we could have the majority of w_{i0} transition to PACU and then from there to the wards using the transition function T.

The waiting demand transitions according to:

$$\vec{y} = \{y_1, ..., y_I\} \rightarrow \left\{ y_i - \sum_{n \in N} b_{in} - D_i^y + Y_i \right\}_{i \in [I]} \qquad (3.15)$$

where D_i^y is a random variable that represents discharges from the wait list for patients of class i, and Y_i represents newly arrived demand.

Finally, the capacity allocation transitions according to:

$$\vec{z} = \{z_{j0}, ..., z_{jN}\}_{j \in [J]} \rightarrow \{z_{j1} + c_{j1}, ..., z_{jN} + c_{jN}, z_{j0} + c_{j0}\}_{j \in [J]} \qquad (3.16)$$

This reflects the desired cyclical nature of the schedule so that day 0 becomes the new day N.

3.3.4 Transition Costs

`

The only remaining piece of information required for the optimality equation is the cost associated with a given state-action pair. Each action comes with a potential cost - a cost for delayed booking, a cost for patients booked past the medically recommended wait time, and a cost for exceeding available capacity, either OR time or bed. Assuming a linear cost structure, we write the costs as:

$$c(\vec{s},\vec{a}) = \sum_{i\in[I]} f^{Delay}(i)\left(y_i - \sum_{n\in[N]_0} b_{in} \right) + \sum_{(i,n)\in[I]\times[N]_0} f^{Late}(i,n) b_{in}$$

$$f^{OverTime}(\vec{w}_0) + \sum_{k\in[K]} f^{Bed}(k)\left[\sum_{(i,n)\in[I]\times[M(i)]} x_{ikn} - C^B(k) \right]^+ + f^{CAP}(\vec{c}_0,\vec{z}_0) \qquad (3.17)$$

where $\vec{c}_0$ and $\vec{z}_0$ represent that capacity allocated for today.

It is intuitively obvious that $f^{Delay}(i)$ will be decreasing in i as it should be more costly to delay the booking of a higher priority patient. However, the actual values are fairly subjective as it is difficult to quantify the impact of a delay on a patient's health.

One reasonable form of $f^{Late}(i,n)$ is:

$$f^{Late}(i,n) = \begin{cases} \sum_{k=1}^{n-T(i)} \gamma^{k-1} f^{Delay}(i), & for\ all \quad n > T(i); \\ 0, & otherwise \end{cases} \qquad (3.18)$$

where $T(i)$ is the wait time target for a patient of class i (Patrick et al. 2008). This insures that the cost of delaying a booking m days and then booking the patient within the medically recommended wait time is the same as initially booking that patient m days late.

`

$f^{OverTime}(\vec{w}_0)$ indicates the overtime costs associated with the day of surgery. The costs associated with carrying a certain amount of capacity, $f^{CAP}(\vec{c}_0, \vec{z}_0)$, is a function of the OR time made available that day.

3.4 Simplified Version of the Markov Decision Process Model for Surgical Scheduling Problem

The size of the state space in the full model is so large that even the ADP methodology may find it intractable. Hence we consider a simplified version of the model without too much complexity in order to determine the appropriate ADP methodology for this problem.

In the data set given to us by the Ottawa Hospital (TOH), there are 900 different surgical procedures performed in the last year, 3 different emergency priority levels and 4 different elective priority levels. The typical block schedule runs for 30 days and bed classes include PACU, ward and ICU. There are 11 different surgical specialties. Not all of these categories can be contained in the model. However, if we simplify the model we can capture the most relevant aspects of the problem. After successfully solving the small sized problem, we can proceed to add more complexity to the model.

The two easiest simplifications are to assume that the block schedule is derived by other means and to assume there is only one type of bed. There are good models available that solve for the block schedule while taking into account available OR time and bed capacity (Santibanez, Begen et al. 2007). In this model we will use the master schedule

`

given to us by TOH. Assuming only one bed type means that we are ignoring the issue of bed capacity in the PACU and simply focusing on ward capacity. With the simplifications stated above we can define model component as:

3.4.1 State Space

The state space reduces to a vector $\vec{s} = \{t, \vec{w}, \vec{x}, \vec{y}\}$ where t represents the current day of the master schedule, w_{in} represents the number of class i patients with surgery in n days, x_{in} represents the number of class i patients who are on their nth day in the hospital and y_i represents the number of class i patients waiting for an appointment.

3.4.2 Action Set

The action set reduces to a vector, $\vec{b}$ where b_{in} represents the number of class i patients to book n days in advance. The capacity decisions are now assumed to be fixed inputs into the model that depend on the value of t. We will continue to represent OR capacity available for surgical specialty j on day n of the master schedule by z_{jn}.

In this version of the model, in order to include the possibility of using overtime into the model, we allow the overtime capacity to be 25th percentile of each surgery length. Since our focus is on elective surgeries and because the hospital is trying to prevent cancellations of elective surgery in order to meet emergency demand, we can remove the

`

constraint associated with emergency booking. As a result, our remaining constraints of the simplified problem would be:

First, the capacity constraint is according to :

$$\sum_{i\in I(j)} l(i)\left(b_{in}+w_{in}\right)\le z_{jt^*} \qquad \forall j\in[J] \qquad (3.19)$$

Second, bookings need to be less than waiting demand:

$$\sum_{n\in[N]_0} b_{in}\le y_i \qquad \forall i\in[I] \qquad (3.20)$$

Third, all bookings are required to be integer and positive:

$$b_{in}\in\mathbb{Z}^+ \qquad \forall(i,n)\in[I]\times[N]_0 \qquad (3.21)$$

Therefore, we can denote the action set for any given state by:

$$A_{\vec{S}}=\left\{(\vec{b},\vec{c})\,|\,\sum_{i\in I(j)} l(i)\left(b_{in}+w_{in}\right)\le z_{jn},\ \sum_{n\in[N]_0} b_{in}\le y_i,(\vec{b},\vec{c})\ge 0,Integer\right\} \qquad (3.22)$$

3.4.3 Fundamental Dynamics

We present the transition of each part of the state vector separately, beginning with the transition of the of current day of the master schedule:

If t is equal to N then we set $t=1$, otherwise set $t=t+1$.Without any change, the transition of the booking schedule is similar to Eq. (3.13):

$$\vec{w}=\{w_{i0},...,w_{iN}\}\rightarrow\left\{w_{i1}-D_{i1}+b_{i1},...,w_{i,N}-D_{i,N}+b_{i,N},0\right\}_{i\in[I]} \qquad (3.13)$$

`

where D_{in}^{w} represents discharges from the booking schedule for patients from class i who are n days from surgery. Since no one is booked more than N days in advance, the N^{th} day is always empty. In order to estimate the probability of discharge from the booking schedule for class i, we take into account the number of booked cases that were canceled in advance.

The hospital census transitions will be according to:

$$\vec{x} = \left\{ x_{i1}, ..., x_{iM(i)} \right\}_{i \in [I]} \rightarrow$$
$$\left\{ w_{i0} - D_{i0}^{x}, x_{i1} - D_{i1}^{x}, ..., x_{i,M(i)-1} - D_{i,M(i)-1}^{x} + x_{i,M(i)} - D_{i,M(i)}^{x} \right\}_{i \in [I]} \qquad (3.23)$$

where D_{in}^{x} represents patients from class i who are discharged after n days. In order to determine the probability of a patient of class i being discharged after n days, we tracked the proportion of patients from class i who were discharged after n days up to a maximum $M(i)$ days. As will be shown in Chapter 4, after forming the patient classes, the discharge probability after $M(i)$ days stabilized to a constant value that could be used for all patients staying longer than $M(i)$ days. Discharge probabilities for all patient classes stabilized to a constant value by 20 days at most.

Ignoring discharges from the wait list, the waiting demand transitions are formulated by:

$$\vec{y} = \{ y_1, ..., y_I \} \rightarrow \left\{ y_i - \sum_{n \in N} b_{in} + Y_i \right\}_{i \in [I]} \qquad (3.24)$$

`

where Y_i is a random variable that represents newly arrived demand for patients of class i .Due to data limitations, we estimated the arrival rate of demand for each surgical specialty. To obtain the arrival parameter for each class within each specialty, the arrival rate for each specialty has been multiplied by the percentage of procedures exit in each class.

3.4.4 Transition Costs

The cost function captures the cost for delayed booking, a cost for patients booked past the medically recommended wait time and a cost for exceeding available capacity – either OR time or bed capacity. Hence we write the costs as:

$$c(\vec{s},\vec{a}) = \sum_{i\in[I]} f^{Delay}(i)\left(y_i - \sum_{n\in[N]_0} b_{in} \right) + \sum_{(i,n)\in[I]\times[N]_1} f^{Late}(i,n) b_{in}$$

$$\sum_{j\in[J]} f^{OverTime}(j)\left(\sum_{i\in I(j)} l(i) w_{i0} - z_{j,t^*} \right)^+ + f^{Bed}\left[\sum_{(i,n)\in[I]\times[M(i)]} x_{in} - C^B \right] \quad (3.25)$$

where in the overtime function t^* is equal to $t+n$ if $t+n$ is less than N ; otherwise t^* equals to $t+n-N$ (to capture the cyclic nature of the block schedule).

The value for bed capacity is also problematic as the number of available beds will change over the year. We chose this value by counting the total number of beds for one month. The same conditions on the remaining costs apply as in the original model.

`

3.5 Summary

In this chapter a comprehensive Markov Decision Process Model has been developed for surgical scheduling problem. Due to the computational difficulty arising because of the enormity of the problem size; the simplified version of the model is proposed. In the next step, the model needs to be populated with the required parameters from a real case. The beginning stage in finding the required parameters is forming the patient classes as most of the parameters are associated with the class of the patient. In the next Chapter we present how the *Clustering* method can be employed for creating patient class and the performance of the created classes will be investigated.

`

Chapter 4: Clustering Technique for Creating Patient Classes

The first challenge to overcome for populating the MDP model with required parameters is to group patients together into a reasonable number of classes. For elective surgeries we need to incorporate the priority category (there are 4 of them), the surgical specialty (there are 9 in the master schedule given to us by TOH), the length of surgery and average for LOS .There are currently 900 different procedure types present in our data. Modeling this many patient classes is impossible and also unnecessary. We need to group these into a more manageable number. Ideally, we would classify patients into a manageable number so that there is no class involving multiple surgical specialties and the distributions of surgical time and LOS are reasonably well-defined within each group.

4.1 Clustering

We used a clustering technique from the data mining set of methods to form the patient classes. Data Mining is a branch of mathematics with the specific intent of gleaning information from large data sets. Clustering is a way of grouping together data samples that are similar in some way - according to a criterion that the user determines.

`

Clustering models focus on identifying groups of similar records and labeling the records according to the group to which they belong. The basic clustering problem distributes data into k different groups such that data points similar to each other are in the same group. Similarity between data points are defined in terms of some distance metric that can be chosen. Clustering is especially useful for similarity and dissimilarity analysis where it analyze what data points in the sample are close to each other. It is also useful for dimensionality reduction with high dimensional data replaced by group (cluster) labels.

4.2 K-means Clustering

K-means clustering is one of the simplest and most popular unsupervised classification techniques. The main difference between supervised and unsupervised classifications is that a training dataset with known class labels is required for the supervised rule, but no training data is required for the unsupervised version with the only input being the number of classes. There are different methods of clustering such as partitioning, hierarchical and so forth. K-means falls into the category of partitioning techniques. In K-means clustering, groups are defined in terms of cluster centre that are also called means. The algorithm minimizes the sum of squared centre-point distances for all clusters.

`

Mathematically, if $Y = [y_1, ..., y_n]$ is a set of data vectors with n representing the number of observations, the K-mean algorithm classifies the data into k clusters that minimize the sum of squares errors, Z :

$$Z = \sum_{i=1}^{K} \sum_{j \in Cluster_i} \left| y_j - \overline{y}_i \right| \qquad (4.1)$$

where $\overline{y}_i$ is the Centre of the cluster i. Therefore the K-means is an iterative algorithm that begins with initializing k values of means or centres, and repeats the following two steps until no changes in the means occurs:

- Partition the data according to the current set of means using the similarity measures
- Move the mean to the centre of the data in the current partition

Some of the advantages of using K-means include simplicity and generality of the algorithm which can work for more than one distance measure. The algorithm always converges (to local optima). However, the result is sensitive to the initial value of the means. As such, it is necessary to run the algorithm several times, each time with a different initialization. The best solution is then taken as the final solution. The drawbacks include the fact that K-means works best for attributes (features) with continuous values and it can perform poorly with overlapping regions.

4.3 Quality of Performed Clustering Methods

`

Since we are interested in managing resources efficiently, it makes sense to group patients based on resource consumption:

- Surgery length
- Post-operative length of stay (LOS)

We considered 9 main surgical specialty services and have clustered the groups within each service based on the 2 attributes. The intended result is that patients in each class have "similar" procedure lengths and post-operative lengths of stay. We have primarily chosen K-means for the purpose of clustering. As stated before, in K-means clustering the user should specify the number of clusters. A number of approaches have been suggested for determining the best number of clusters. Here, the number of clusters is determined by assessing the quality of the final solution over the range of values for k .We have performed K-mean clustering for a range of 2 to 10 clusters and chose the result with best quality. The quality measure here is used by the software is "*Sihouette measure of cohesion and separation*[1]". In addition to K-means, we also tested a method called *Kohonen* which automatically determines the number of clusters. It had considerably lower quality compared to K-means. The result for both methods is reported below in Figure 4 to Figure 7. It can be seen from the "Cluster Quality" graph that K-mean method with 3, 4 and 5 clusters has "Good" quality as opposed to Kohenon method with "Fair" Quality and 12 clusters. You can also see in the K-mean method, "Quality" decreases as we increase the number of clusters from 3 to 4 and 5 which indicate that 3

[1] Explanation of this quality measure is out of scope of this thesis.

`

clusters is the preferred number of clusters in this case. In the following, we present the results for thoracic service. This procedure for other services is similar. *IBM SPSS Modeller* and *Rapid Miner* software is used for the purpose of executing the clustering technique

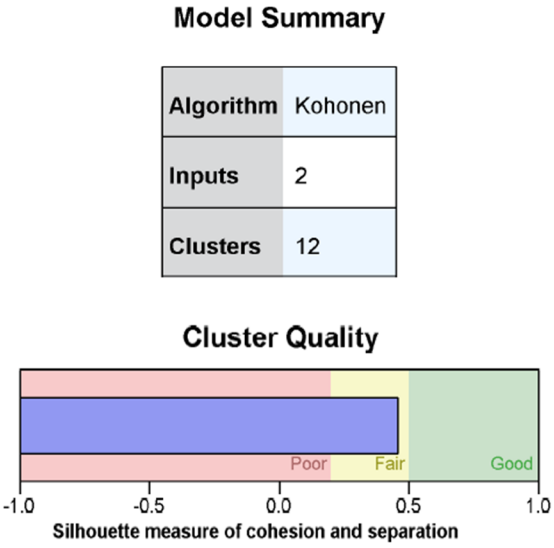


Figure 4. Cluster Quality for Kohonen Algorithm

`

Figure 5. Cluster Quality for K-Means Algorithm with 3 Clusters

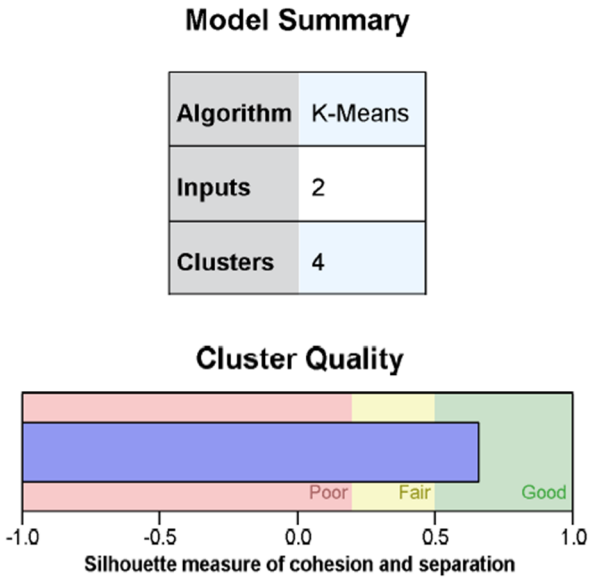


Figure 6. Cluster Quality for K-Means Algorithm with 4 Clusters

`

Model Summary

Algorithm	K-Means
Inputs	2
Clusters	5

Cluster Quality

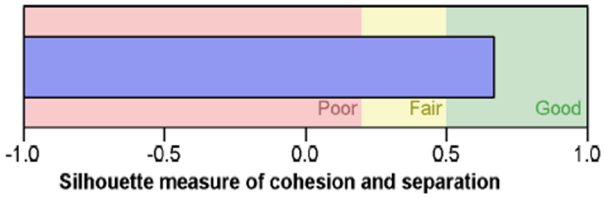
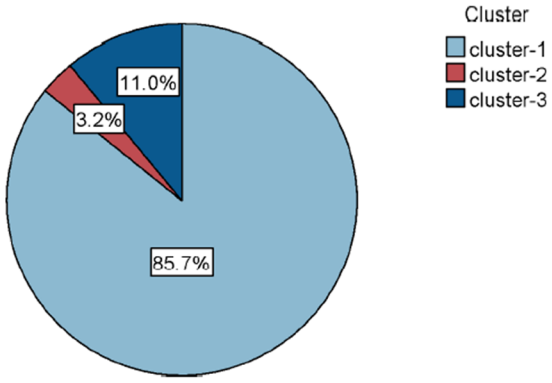


Figure 7. Cluster Quality for K-Means Algorithm with 5 Clusters

Figure 8 shows the result of "cluster sizes" graph for thoracic service.

Cluster Sizes



Size of Smallest Cluster	20 (3.2%)
Size of Largest Cluster	528 (85.7%)
Ratio of Sizes: Largest Cluster to Smallest Cluster	26.40

Figure 8. Cluster Sizes Graph for thoracic Service

`

The resulting “Cell distribution” for length of stay (LOS) attribute for each group is demonstrated in Figure 9 to 11. In these in following Figures, X axis represent length of stay in hospital based on days and Y axis is the frequency of cases.

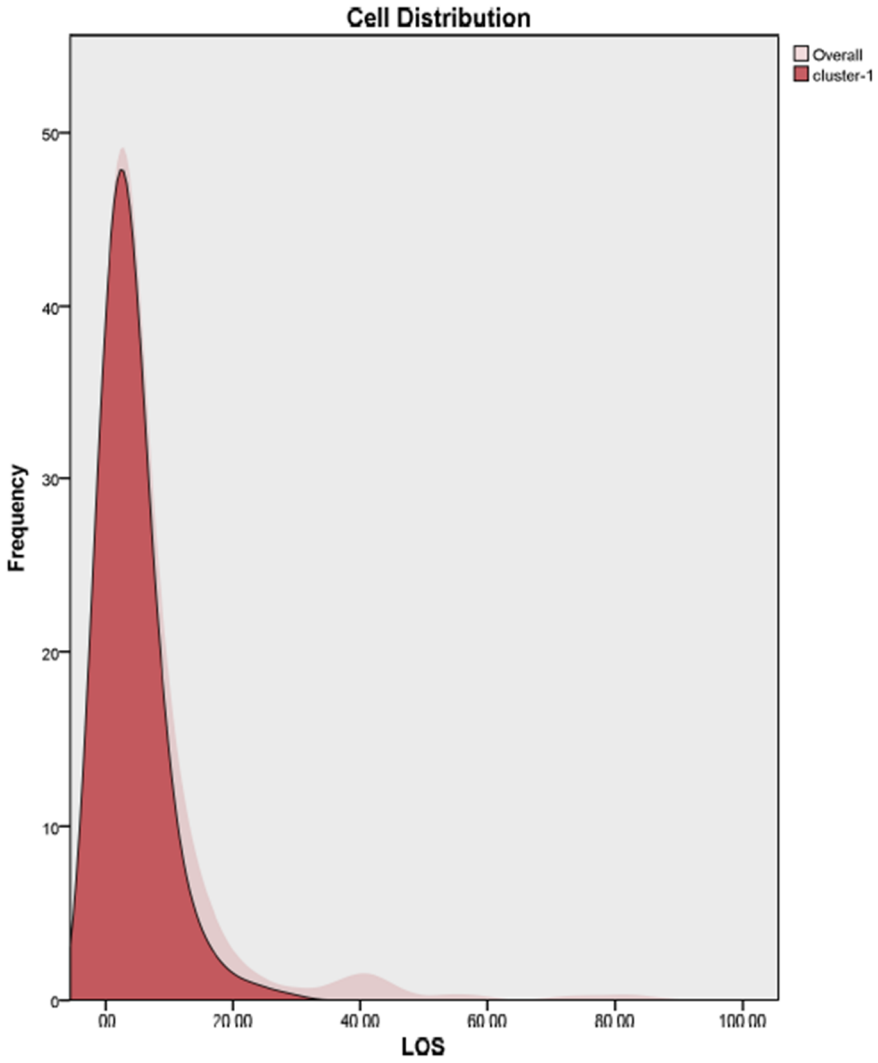


Figure 9. LOS Cell Distribution for Thoracic Service Class 1

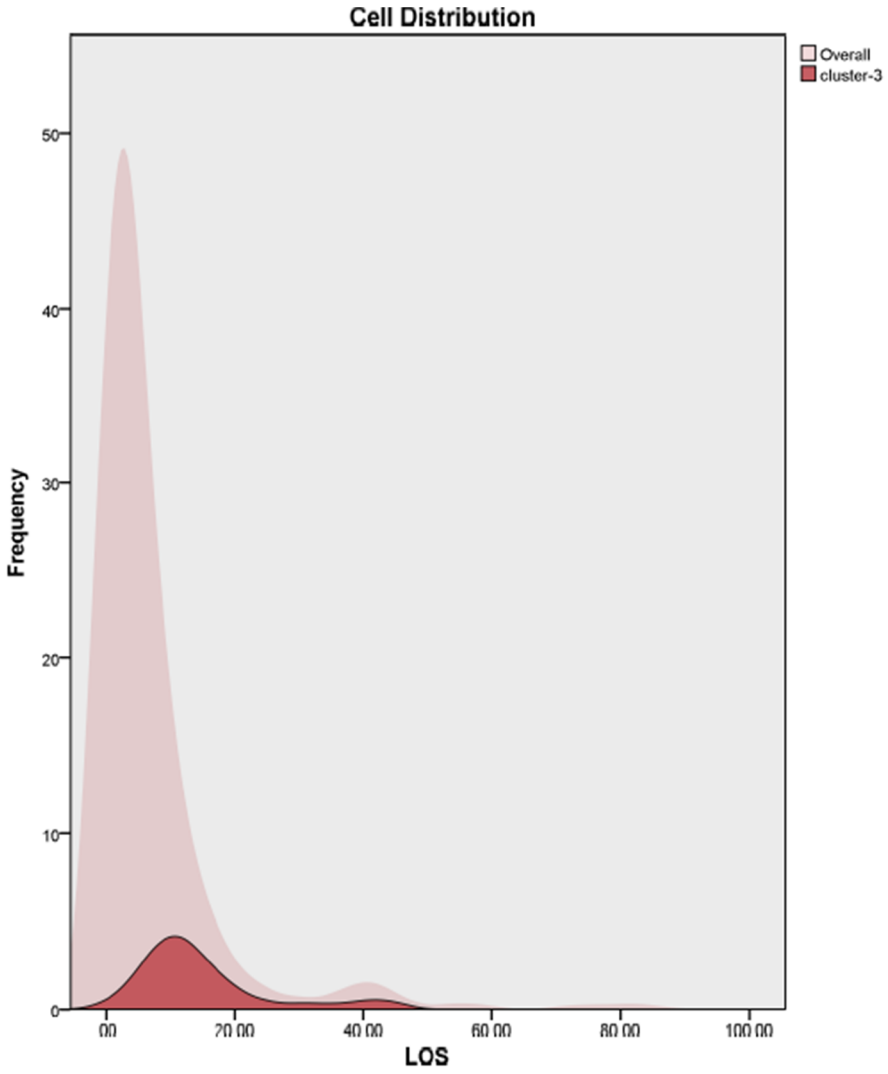


Figure 10. LOS Cell Distribution for Thoracic Service Class 2

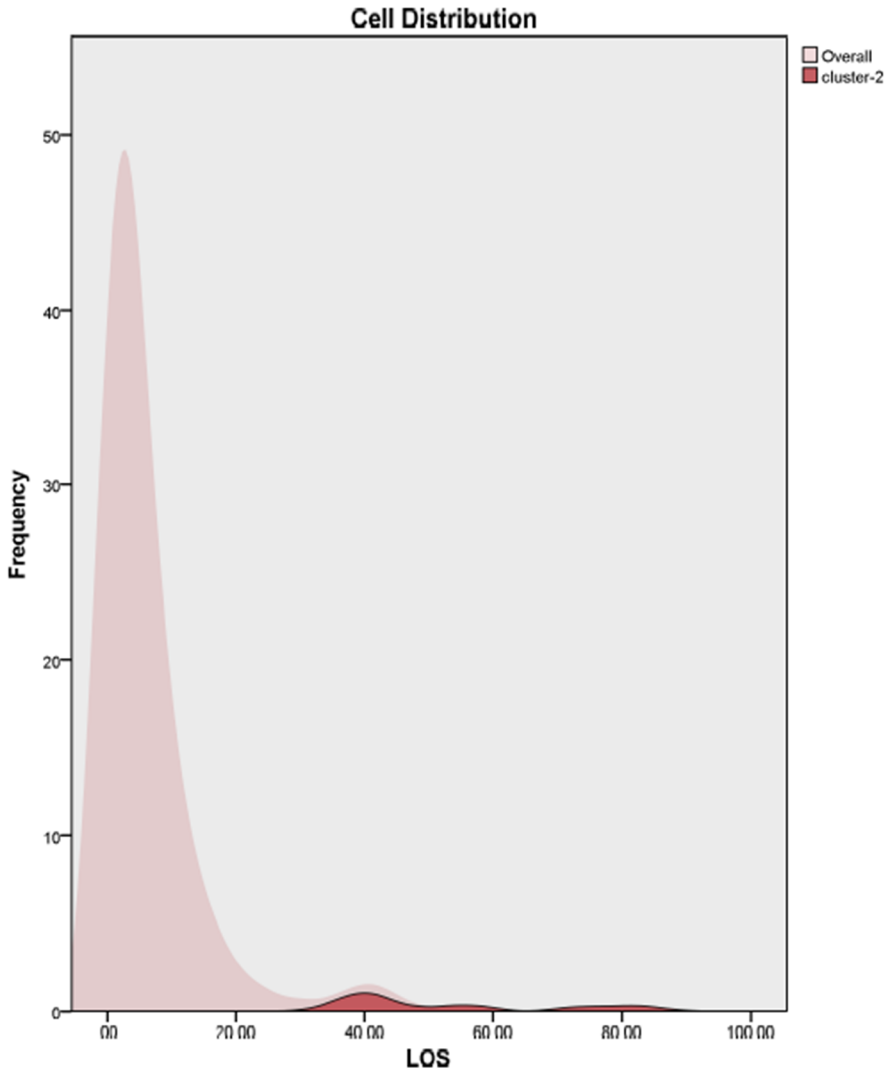


Figure 11. LOS Cell Distribution for Thoracic Service Class 3

Scatter 3D plot for thoracic service representing clusters are formed based on 2 attributes is illustrated in Figure 12. The X axis represent the length of stay attribute, the Y axis represents LOS attribute and the Z axis represents clusters. The scatter 3D plot shows dependencies between the three dimensions.

`

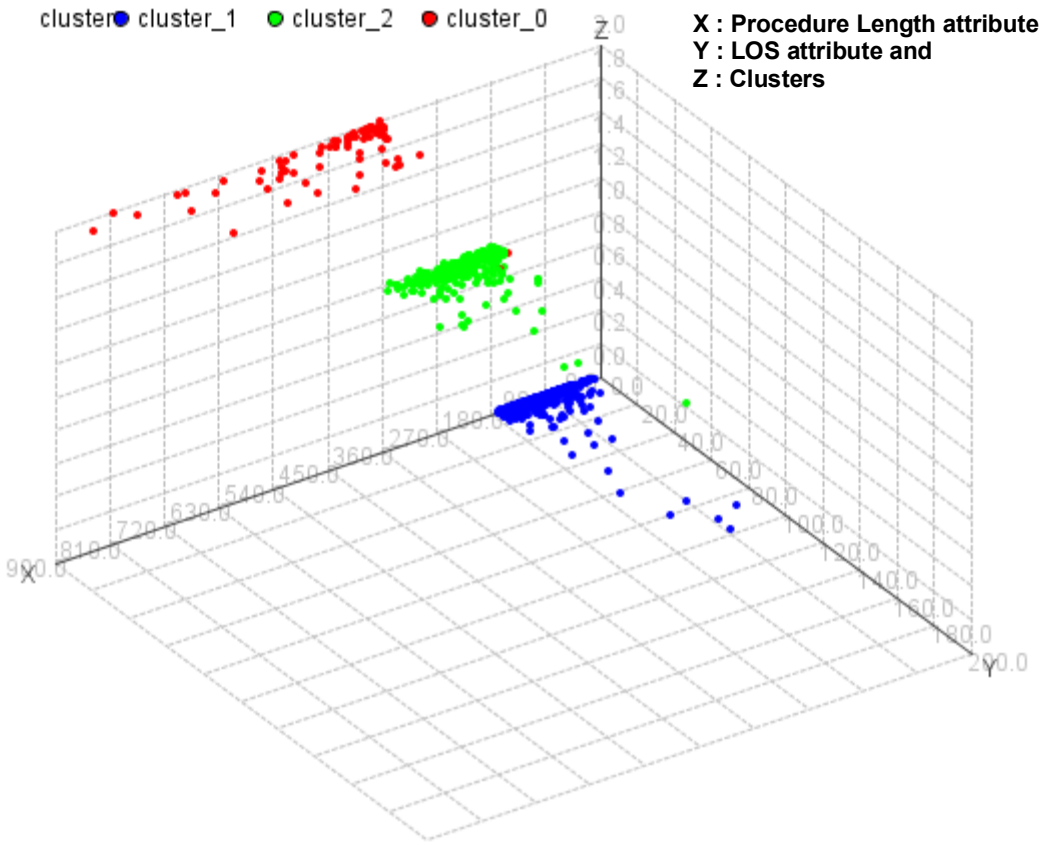


Figure 12. 3D Scatter Plot of Thoracic Service Clusters Distribution

2D version of the scatter plot for thoracic service is also pictured in Figure 13 which is again plotted base on the two attributes LOS and Surgery Length. In this graph, X axis represents surgery length attribute based on minute and Y axis represents length of stay attribute in based on days.

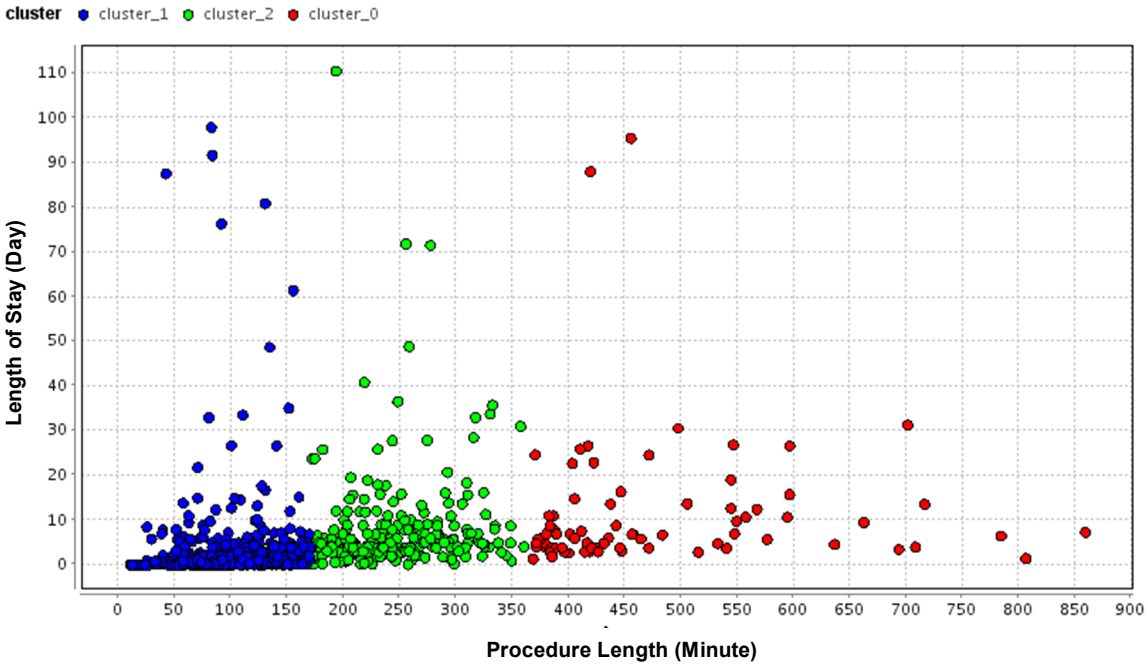


Figure 13. 2D Scatter Plot of Thoracic Service Clusters Distribution

4.4 Resulting Distributions

After performing a clustering analysis, we obtained three classes per specialty with surgical procedures grouped based on surgical time and length of stay. The first class can be roughly interpreted as a class for surgeries with short length, second class with medium surgery length and third class with long surgery length.

Once the classes are determined, it is possible to begin to determine how these classes should be scheduled based on their expected resource consumptions and associated wait time targets. It is crucial to capture from the historical data the following three distributions for each patient class:

- Distribution of demand (number of new requests for surgery per day)

`

- Distribution of procedure length (number of minutes per surgery)
- Distribution of LOS (number of days per patient post-surgery)

Once these are known, we would be able to populate our MDP model with the necessary parameters. Finding the best fitting distribution that matches our data has been carried out using *EasyFit* software which fits the best distribution among more than 40 distributions available in its repository. It ranks fitted distributions based on the "Anderson Darling", "Kolmogorov Smirnov" and "Chi-square Goodness of Fit" tests.

The distribution of demand for most of the groups follows a Poisson distribution and for the remaining groups is very close to Poisson. We therefore use the Poisson for all classes to estimate the demand arrival parameters. Figure 14 shows the distribution of demand for neurology service class 1 as an example.

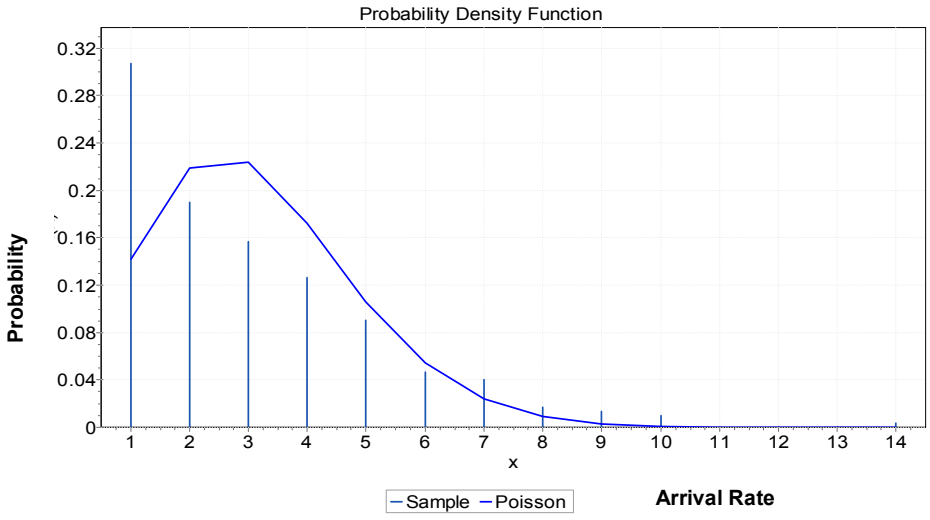


Figure 14. Distribution of Demand for Neurology Service Class 1

`

The distribution of procedure length in group one and two for each specialty follows a lognormal distribution. Lognormal distributions have been shown in the literature to be the most common distribution for best estimation of surgery length. For example, Strum, et al., (2000) illustrated that the log-normal model is superior to the normal model for large sets of surgery times. Therefore, the practice of estimating surgery lengths based on a lognormal model is widely adopted. A log-normal distribution has positive support and positive skewedness, which is appropriate to surgery times in that a few cases may take much longer than average. For instance Figures 15, 16 and 17 show the distributions of procedure length for neurology service for class 1, 2 and 3 respectively. As it illustrates, class 1 and 2 are clearly following log-normal but in the case of the third class, the distribution is slightly skewed to the left which makes it to get closer to Weibull and Normal distribution.

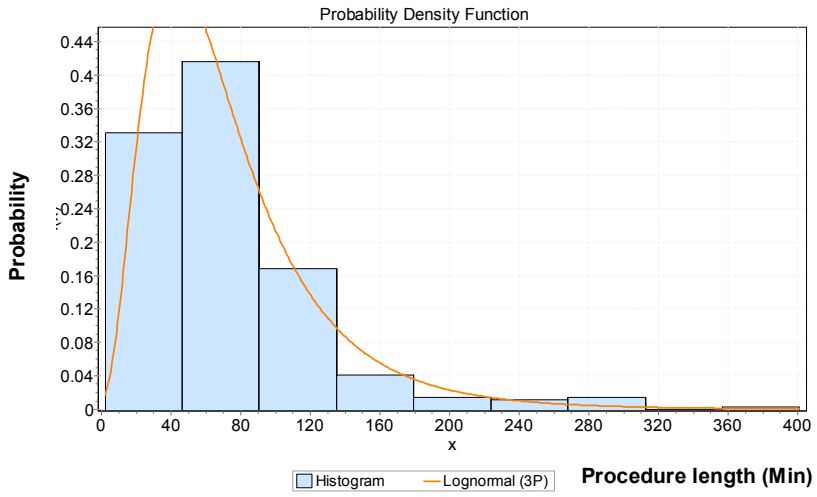


Figure 15. Distribution of Procedure Length for Thoracic Service Class 1

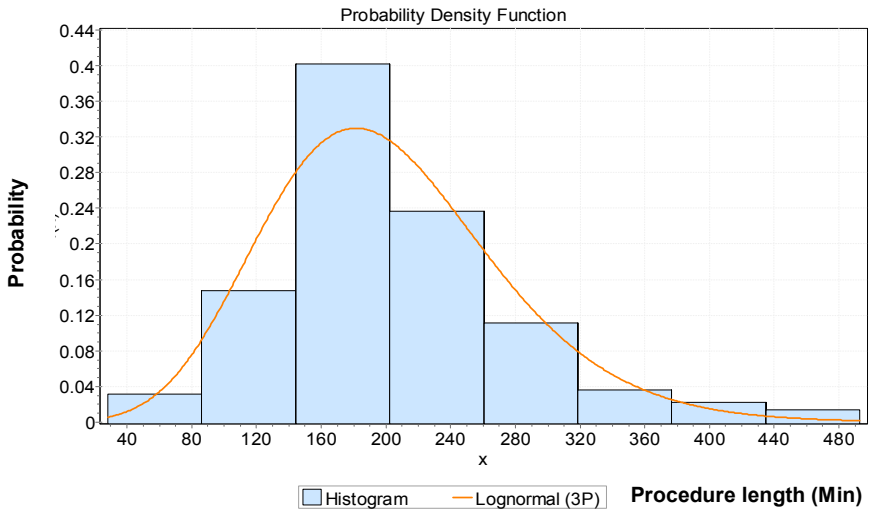


Figure 16. Distribution of Procedure Length for Thoracic Service Class 2

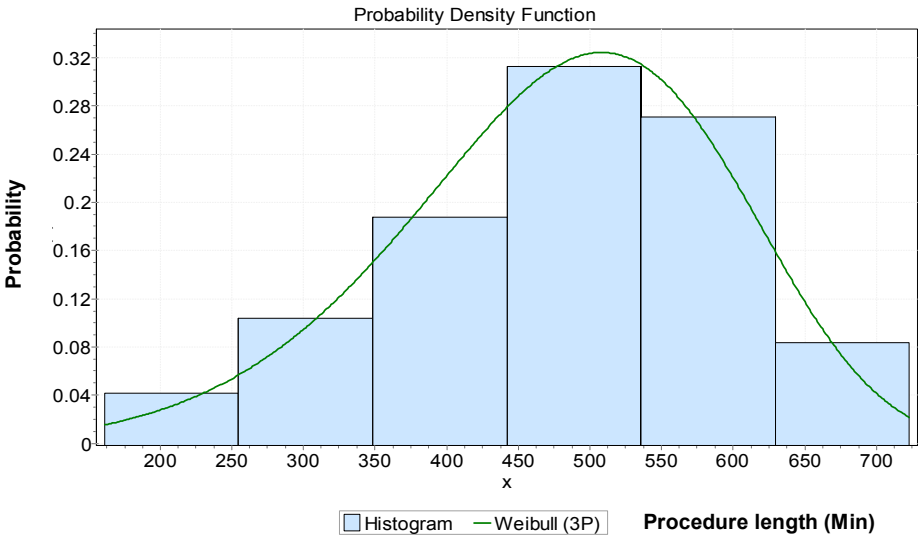


Figure 17. Distribution of Procedure Length for Thoracic Service Class 3

We also need to determine the length of stay (LOS) in the hospital for each of our patient classes. In particular, we need to determine the probability of a patient of class *i* being discharged after *n* days. Intuitively it should stabilize after a period of time so that

`

we can simply track LOS for that period and then have a category for patients who stay longer than that. Thus, the vector, $\vec{x}$, will have 27 patient categories as well. The values of $M(i)$ as the maximum length of stay in the hospital for patient class i, are based on that point in the data when the probability stabilizes. For instance Figure 18 shows the distribution of length of stay for neurology Service Class 2. It clearly shows that it stabilizes to a constant value after 20 days.

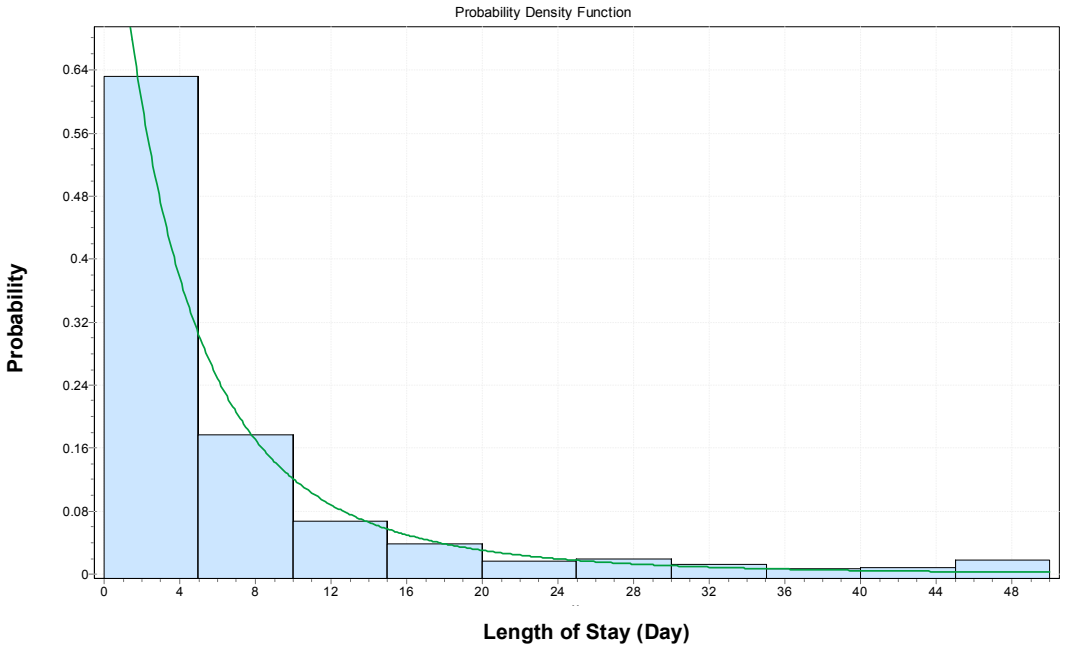


Figure 18. Distribution of Length of Stay, Neurology Service Class 2

4.5 Summary

`

In this chapter, the clustering approach from the set of data mining methods has been established for creating the patients classes. After forming patient classes, we were able to determine the necessary parameters for each class in order to populate the MDP model. Now that we have the MDP populated with parameters, the next step would be solving the MDP model using the appropriate methodology form the field of Approximate Dynamic Programming. Hence the next Chapter deals with investigating the appropriate ADP methodology that best suites our MDP model and with implementing such a methodology to determine a solution.

`

Chapter 5: Simulation-Based Approximate Dynamic Programming for the Surgical Scheduling Model

In this section we provide a background of traditional methods for solving an MDP as well as approximate dynamic programming (ADP) approach to solve the MDP. As stated in Chapter 3, for solving a MDP model, the primary interest is the policy μ^* which satisfies the minimum in equation (3.1). As also mentioned in Chapter 3, the optimal value function associated with μ^*, indicated by J^* will satisfy the Bellman optimality equations:

$$J^*(i) = \underset{u \in U}{Min}\left(C(i,u) + \gamma \sum_{j \in S} P_{ij}(u) J^*(j) \right) \qquad \forall i \in S \qquad (3.5)$$

And as a result if J^* can be found by solving equation (3.5), then the optimal policy μ^* can be found using equation (3.6):

$$\mu^*(i) = \arg \underset{u \in U}{Min}\left(C(i,u) + \gamma \sum_{j \in S} P_{ij}(u) J^*(j) \right) \qquad (3.6)$$

Traditional dynamic programming algorithms such as *value iteration* and *policy iteration* (Bellman 1954) can be used to determine J^* and the optimal policy μ^* for

`

small-sized MDPs (Howard 1960). One approach for computing the optimal value functions is to solve equation (3.5) in an iterative fashion using a procedure known as *value iteration* (Bellman 1954). Once the value function is known, the associated policy can be computed by equation (3.6).

Another method known as *policy iteration* can be applied to solve small size problem directly (Bellman 1954). Because our main solution methodology is an extension of the classical policy iteration algorithm, we provide a brief description of this method in the following section.

5.1 Policy Iteration

In the policy iteration algorithm we start with an initial policy μ_0. The initial policy is usually chosen to be some reasonable heuristic. Given the policy μ^k the value function of each state $i \in S$, $J^{\mu k}(i)$ can be obtained by solving the following linear system of equations:

$$J^{\mu}(i) = C\left(i, \mu^k(i)\right) + \gamma \sum_{j \in S} P_{ij}\left(\mu^k(i)\right) J^{\mu k}(j) \qquad \forall i \in S \qquad (5.1)$$

where the $J(i)$'s are unknown. This step is called the *policy evaluation*. Once the value functions $J^{\mu k}(i)$ have been determined, the *policy improvement* is performed to obtain the next policy μ^{k+1} according to:

`

$$\mu^{k+1}(i) = \arg \underset{u \in U}{Min} \left(C(i,u) + \gamma \sum_{j \in S} P_{ij}(u) J^{\mu k}(j) \right) \qquad \forall i \in S \qquad (5.2)$$

By iteratively performing policy evaluation followed by policy improvement, a sequence of policies is obtained that is guaranteed to converge to the optimal policy μ^*. The algorithm is guaranteed to converge in a finite number of iterations and to generate a sequence of proper policies so that $J^{\mu_{k+1}}(i) \leq J^{\mu_k}(i)$ (Bertsekas and Tsitsiklis 1996) .For sufficiently large i, and for problems with a finite number of policies, μ_i will converge to μ^* and J^{μ_i} will equal to J^*. This method however becomes computationally infeasible for problems with large state spaces because we need to compute and store the value function $J^{\mu k}$ for each state.

5.2 Approximate Dynamic Programming

Unfortunately, to determine optimal policies for realistic sized systems, the MDP model becomes challenging. For real problems with large state spaces, it is not possible to apply traditional dynamic programming methods such as policy iteration directly because of the long computation time and memory requirements. This is referred to as *the curse of dimensionality* (Powell 2007).

The curse of dimensionality refers to the fact that when the problem size increases, the size of the state space, and therefore the amount of computation necessary to solve the Bellman optimality equation grows exponentially. For typical problems, with this

`

exponential growth the determination of optimal policies becomes infeasible because of the long computation times and large memory requirements making it difficult or even impossible to solve the problem exactly in reasonable time. Our problem suffers from the curse of dimensionality even after creating patient classes. In particular the dimension of the state space can be calculated by $C^{IN}D^{IN}M^{I}$, where C is the maximum number of patients that can be booked within each block , D is the maximum number of patients in hospital and M is the maximum number of patients from single class who arrive on a given day. By replacing these parameters with some reasonable number, the state space dimension would be roughly around $(6^{27*20})(5^{27*20})(4)$ which is a huge number.

Another difficulty that may arise is called the *curse of modeling* which refers to the difficulty of modeling the dynamics of the system (Gosavi 2003). Many systems exhibit uncertainty in their parameters. In this case, the basic character of the dynamic equations is known, but the exact values of one or more parameters are not.

The question of how to address these challenges and how to overcome these two curses has been the subject of a great deal of research, leading to the development of fields known as Approximate Dynamic Programming, reinforcement learning, and neuro-dynamic programming that provide methodologies for solving larger MDP models (Bertsekas and Tsitsiklis 1996; Sutton and Barto 1998). Semantically, these fields are essentially equivalent (having different names due only to the fact that they were originally developed by different communities), and for the sake of clarity, we refer to all of these methods together as approximate dynamic programming (abbreviated ADP).

`

ADP is comprised of a set of algorithms for solving large scale MDPs. Here we outline some aspects of ADP methodology for solving otherwise intractable MDPs. In a vast range of practical applications, finding the one-step transition probability matrix $P_{ij}(.)$ required to compute the expectations is computationally intractable. In order to circumvent this problem we introduce the *Pre- decision* and *Post-decision* state variables. This strategy has proven to be effective in overcoming the computational challenges around evaluating the expectation. The pre-decision state variable, represented by i, refers to the information required to make a decision u , while the post-decision state variable, i', refers to the state of the system immediately after a decision u is made. With the help of the post-decision concept, we can break our original transition function into two steps, given by:

1) $i' = f_u(i,u)$ representing the pure effect of a decision u and 2) $i = f_w(i', w)$ representing the effect of the exogenous information w. Exogenous information refers to a set of random variables representing the information that arrives after making decision u and that changes the state of the system from the post-decision to the pre-decision state.

In our surgical scheduling model the exogenous information consists of the Poisson arrivals of new demand, discharges from the booking scheduling and discharges from the ward that occur between the end of the current day t and the start of a new day $t+1$. Hence, having made booking decision b_{in}, the post-decision would be:

`

$$\vec{w} = \left\{ w_{i1} + b_{i1}, \ldots, w_{i,N} + b_{i,N} \right\}_{i \in [I]} \qquad (5.3)$$

$$\vec{x} = \left\{ w_{i0}, x_{i1}, \ldots, x_{i,M(i)-1} + x_{i,M(i)} \right\}_{i \in [I]} \qquad (5.3)$$

$$\vec{y} = \left\{ y_i - \sum_{n \in N} b_{in} \right\}_{i \in [I]} \qquad (5.5)$$

The transition from post-decision to pre-decision occurs after the arrival of new stochastic information according to:

$$t \to t+1 \qquad (5.6)$$

$$\vec{w} = \left\{ w_{i1} - D_{i1}, \ldots, w_{i,N} - D_{i,N}, 0 \right\}_{i \in [I]} \qquad (5.7)$$

$$\vec{x} = \left\{ w_{i0} + D_{i0}, x_{i1} - D_{i1}, \ldots, x_{i,M(i)-1} - D_{i,M(i)-1} + x_{i,M(i)} - D_{i,M(i)} \right\}_{i \in [I]} \qquad (5.8)$$

$$\vec{y} = \left\{ y_1, \ldots, y_I \right\} \to \left\{ y_i + Y_i \right\}_{i \in [I]} \qquad (5.9)$$

Schematic representation of the system's evolution using the pre and post-decision idea is illustrated in Figure 19:

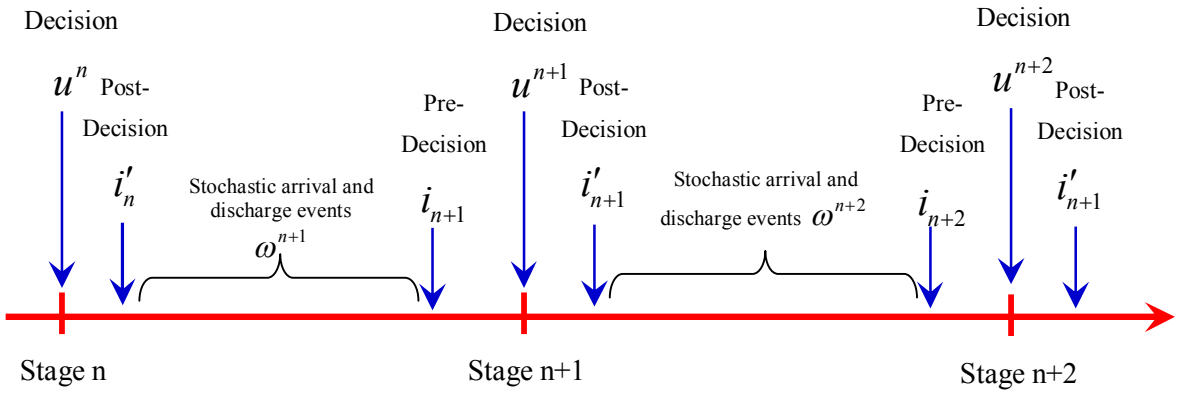


Figure 19. ADP System's Evolution Over Time

`

In this way we are able to drive deterministic optimality equations around the post decision state variable given by

$$J(i) = \underset{u \in U}{Min}\left(C\left(i,u\right) + \gamma J(i')\right) \qquad (5.10)$$

This is the primary advantage of the post-decision framework along with a reduction in the size of the state space. This way instead of having known all the value functions for all the possible states that could be reached from state i, the value function just needs to be known for the immediate post-decision state $J(i')$.

On the other hand, because of the curse of dimensionality, for most practical problems we would not be able to compute $J(i')$ even for every post-decision state. A central idea for addressing the curse of dimensionality is the use of a function approximation architecture to represent the value function at the cost of losing precision. The fundamental idea is to assume that the value function has a functional form that can be characterized by a set of parameters similar to that of statistical regression. Thus, instead of computing the value function for every state $i \in S$, ADP methods use a parameterized class of functions $\tilde{J}\left(.,r\right)$ to approximate J^{μ} and the optimal value function J^{*}. In particular for state i, we approximate $J^{\mu}\left(i\right)$ and the optimal value function $J^{*}\left(i\right)$ by a suitable approximation architecture $\tilde{J}\left(i,r\right)$ and then compute a vector of tunable parameters $\vec{r}$ to fit the optimal value function so that: $\tilde{J}\left(i,r\right) \approx J^{*}$. Thus instead of presenting a look-up table that provides the optimal action for each given state, the ADP seeks to provide the optimal parameter values $\vec{r}$ in order to get the best

`

approximation possible. This will lead to a set of optimality equations that can be used to determine the "optimal" action. It has been described in more details in the following section.

5.2.1 Form of Approximation Architecture and Basis functions

In general choosing the appropriate approximation architecture is somewhat dependent on the nature of the problem. The type of approximation architecture employed is a key property of any ADP method and is very important for the success of the approximation approach. An inappropriate choice of approximation architecture would result in terrible outcomes.

In this thesis we consider a parameterized class of functions known as a *linear architecture* of the form $\tilde{J}(i,r)=\sum_{p=1}^{P} r_p \varphi_p(i)$, where $\vec{r}=\left(r_1,...,r_p\right)$ is the vector of tunable parameters and $\vec{\varphi}(i)=\left(\varphi_1(i),...,\varphi_p(i)\right)$ is the vector of fixed Basis functions at state i, also known as the *feature vector* of state i. This methodology involves two stages: a "*feature extractor*" and a "*function approximator*" (See figure 20)(Bertsekas and Tsitsiklis 1996).

`

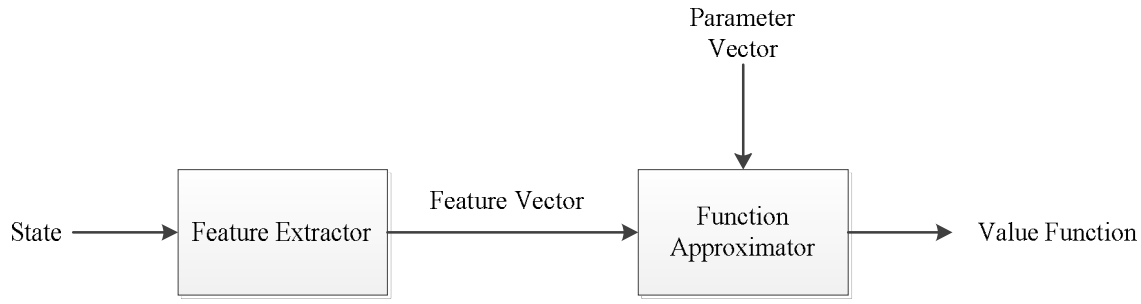


Figure 20. A Feature-based Approximation Architecture

The feature extractor uses the post-decision state i' to compute the feature vector $\vec{\varphi}(i')$.This feature vector is then used as input to a generic function approximator parameterized by a vector $\vec{r}$. Therefore for every state i, the approximate value function $\tilde{J}(i,r)$ is the inner product of $\vec{\varphi}(i)\vec{r}$. Choosing a good set of basis functions remains a challenge within ADP. The elements of $\vec{\varphi}(i')$ are values that should capture the key information concerning states of the system under study.

A reasonable starting point in our model based on the above linear architecture is the following *affine* approximation:

$$\sum_{p=1}^{P} r_p \varphi_p = r_0 + \sum_{i \in I} r_i^w w_i + \sum_{i \in I} r_i^y y_i \qquad (5.11)$$

We choose the basis functions to be the exact value of the number of patients already booked from class i all over the booking horizon, $w_i = \{w_{i0}, ..., w_{iN}\}$, and the number of patients waiting to be booked from each class y_i. One parameter is associated to w_i, r_i^w and one parameter to y_i by r_i^y. The role of the r_0 here is similar to allowing a regression

`

equation not to go through zero. This simple approximation can be easily interpreted. r_i^w represents the marginal infinite-horizon discounted cost of occupied OR time by patients from class i, and r_i^y represents the marginal infinite horizon discounted cost of having one more patient of class i waiting to be booked. One of the main reasons of using a linear architecture is that the approximate policy iteration algorithm is guaranteed to converge (Powell 2010).

Once the approximation architecture is known, we can replace the original value function with the approximation $\bar{J}(i')$ and solve:

$$J(i) = \underset{u \in U}{Min}\left(C\left(i,u\right) + \gamma \bar{J}(i')\right) \qquad (5.12)$$

Solving the optimality equation using the obtained approximation outlines the near optimal action for each given state. As a result of the above affine approximation, the Bellman optimality equation will turn to:

$$u_t = \arg\underset{u \in U}{Min}\left\{c(i_t,u) + \gamma\left(r_0 + \sum_{i \in I} r_i^w w_i + \sum_{i \in I} r_i^y y_i\right)\right\} \qquad (5.13)$$

Therefore, the resulting optimality equation can be solved subject to the constraints to determine the optimal action at each stage of the MDP model.

5.2.2 Solving the ADP

`

Once the functional form has been chosen, there are two main streams of ADP methods that can be employed for determining optimal values for the tunable parameters: the linear programming based approach and the simulation based approach.

In the linear programming based approach, the general idea is to transform the MDP model into the equivalent linear programming (LP) version of Bellman's optimality equations and use the approximate value function which makes the LP model more tractable.

The second method is the simulation–based approach which finds an approximate solution to the Bellman equation by simulating the evolution of the system over a number of initial states in order to tune the parameters (Bertsekas and Tsitsiklis 1996; Sutton and Barto 1998). Simulation-based solutions generate sample paths of the problem and seek to update the parameters that determine the chosen form of the approximation architecture in an iterative fashion. In such methods the value function is approximated and a further source of approximation exists through sampling error.

Both classes of these algorithms are aimed at finding a set of parameters so that $\tilde{J}(.,r)$ gives a good approximation to the value function, J^*.

5.2.3 Some Background

Perhaps two of the notable studies in the area of linear approximation are the works by de Farias and van Roy (2003) and Adelman (2005).

`

de Farias and van Roy (2003) use an efficient approach that "fits" a linear combination of pre-selected basis functions to the value function. They develop error bounds that characterize the quality of the approximations produced by the linear programming approach and the quality of the policy ultimately generated. In general, approximate linear programming involves the solution of linear programs with few variables but an intractable number of constraints. They propose a constraint sampling scheme that retains only a tractable subset of the constraints. They finally show that the resulting approximation is comparable to the solution that would be generated if all constraints were taken into consideration.

Adelman (2005) develops a deterministic linear program (LP) for bid-price control and uses a functional approximation to the value function that is linear in the state variables. He employs a column generation procedure to solve the LP to within a specified optimality tolerance. His numerical results outline economic and computational performance.

A successful example of applying the linear programming based approach is the work by Patrick et al. (2008). They address a patient scheduling problem for diagnostic resources and formulate the problem as a discounted infinite-horizon Markov decision process (MDP) and transform the MDP into the equivalent linear program (LP). They solve the approximate linear problem through column generation on the dual to derive an estimate of the value function in the MDP.

`

There are a number of different methods for solving the simulation based approach that evolved significantly over the past few years. Here we only present a couple of the most important methods that form the basis for many of the other variants. One of the earliest methods is Q-learning developed by Watkins (1989). The central idea is to consider a $Q(s,a)$ factor for each state action pair and during learning, the algorithm assumes the old value and updates the Q factors based on the new information according to the following formula:

$$Q(s_t,a_t) \leftarrow$$
$$Q(s_t,a_t)(1-\alpha(s_t,a_t))+\alpha_t(s_t,a_t)\times\left(Cost_{t+1}+\gamma\max_{a_{t+1}}Q(s_{t+1},a_{t+1})-Q(s_t,a_t)\right) \quad (5.14)$$

where α is the learning rate and γ is the discount factor. By storing a factor table for each state action pair, Q-learning loses its tractability when the system becomes more complex, so we turn to function approximations to come to a solution.

The other important set of methods is based on temporal difference learning (TD) initially developed by Sutton (1988). Temporal difference (TD) learning is an approach to learning how to predict a quantity that depends on future values of a given signal. The prediction at any given time step is updated to bring it closer to the prediction of the same quantity at the next time step. In the context of value function approximation, during the policy evaluation stage, for given policy μ, the simplest form of TD method updates the value functions according to:

$$J(i_t') \leftarrow J(i_t')+\alpha\left(C\left(i_t',u\right)+\gamma J(i_{t+1}')-J(i_t')\right) \quad (5.15)$$

`

The value function can be replaced by any approximation architecture choice. Convergence of temporal difference methods is studied by Tsitsiklis and Van Roy (1997). The least squares temporal difference (LSTD) learning algorithm is the combination of the TD algorithm with a least squares approximation of the value function and was developed by Bradtke and Barto (1996). They prove convergence for the algorithm when it is used with a function approximation that is linear in adjustable parameters. They define a recursive version of the LSTD algorithm (RLSTD) and provide a simulation experiment showing significant improvement in the learning rate achieved by RLSTD.

Tsitsiklis and Van Roy (1996) developed a methodological framework representing value functions using features. They developed algorithms that employ a feature based compact representation that involves feature extraction and a simple approximation architecture. They prove the convergence of the algorithm.

Simulation based ADP has been successfully applied to a number of applications including call–admission control problems(Marbach, Mihatsch et al. 2000), inventory management, (Van Roy, Bertsekas et al. 1997), retailing (Simester et al. 2006), and finance(Tsitsiklis and Van Roy 2001).

5.2.4 ADP Approach for Our MDP Model

As previously mentioned, our MDP model suffers from the curse of dimensionality because of the enormity of the state space which makes the model intractable by standard

`

methods and thus we turn to use the methods of Approximate Dynamic Programming (ADP). In this thesis we consider a general class of simulation based methods called approximate policy iteration (Bertsekas and Tsitsiklis 1996) to solve the underlying MDP model. In particular we use the approximate version of policy iteration based on the Monte Carlo simulation to tune the approximation parameters.

We begin with a description of the approximate version of the policy iteration method provided in section (5.1) and then we present the simulation based version of this method which is actually used as a solution methodology in this thesis.

5.3 Approximate Policy Iteration

We discuss here a generic version of approximate policy iteration. Approximate policy iteration is a generalization of classical policy iteration. As mentioned in before, when the state space becomes very large it is not possible to directly apply classical policy iteration because in every stage k, we need to compute and store value function over the policy μ at stage k, $J^{\mu k}$ for every single state in the state space. This would make the problem computationally infeasible. The general structure of Approximate Policy Iteration is similar to the classical policy iteration. Again, we produce a set of improving policies $\mu^k : S \to U$. However, instead of computing the value function exactly at each stage for every state, the value function $J^{\mu k}$ of the policy μk is replaced

`

by some architecture $\tilde{J}(.,r_k)$, where r_k is vector of tunable parameters chosen to make $\tilde{J}(.,r_k)$ close to $J^{\mu k}$. As a result, the next policy is obtained according to:

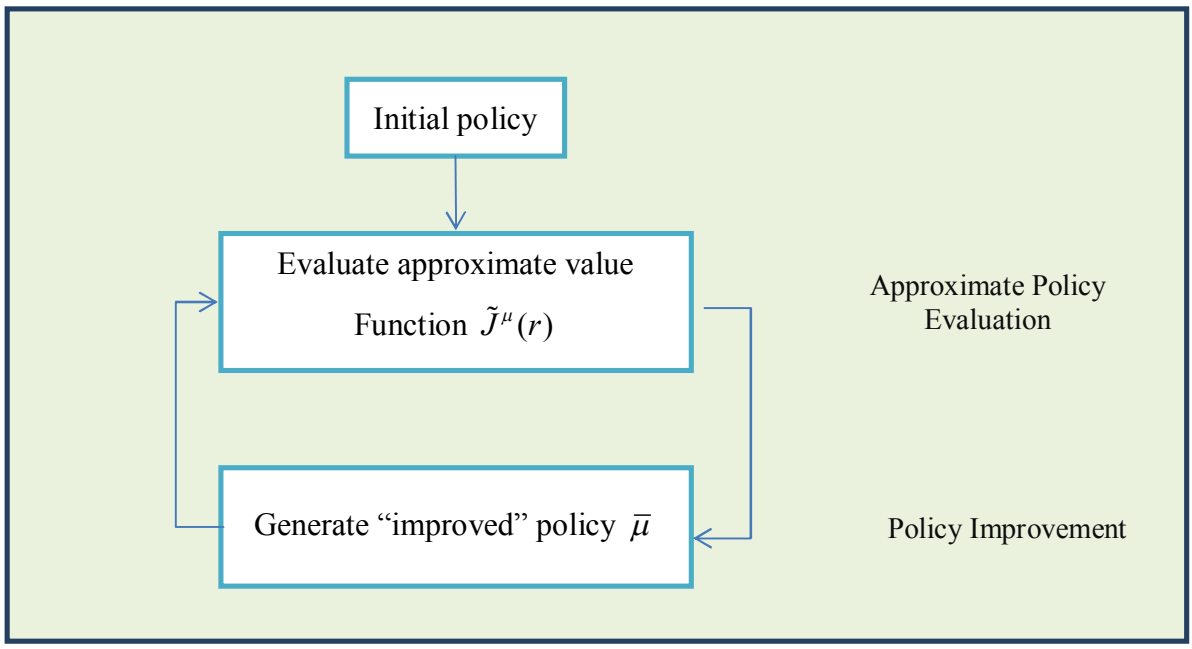
$$\mu^{k+1}(i) = \arg\min_{u \in U}\left(C(i,u) + \gamma \sum_{j \in S} P_{ij}(u)\tilde{J}(j,r_k) \right) \qquad (5.16)$$

Using the concept of post-decision state, we can rewrite equation (5.16) as:

$$\mu^{k+1}(i) = \arg\min_{u \in U}\left(C(i,u) + \gamma \tilde{J}(f_u(i',u),r_k) \right) \qquad (5.17)$$

Figure 21 illustrates the block diagram of generic approximate policy iteration. Similar to classical policy iteration, in approximate policy iteration there is typically an outer loop called policy improvement along with an inner loop where policy evaluation is run for a fixed policy. The policy improvement loop is somewhat standard but there are many variations of the policy evaluation algorithm. Once the policy has been fully evaluated and $\tilde{J}^{\mu}(r)$ is available, the new policy is then generated according to equation (5.16) or (5.17). Similar to policy iteration, in order to run the approximate policy iteration a good starting policy is of paramount importance. Bertsekas and Tsitsiklis (1996) prove that the approximate policy iteration guaranteed convergence when it is used in a finite state space.

`



Source: "Dynamic Programming and Optimal Control", 3rd Edition, Volume II, by Dimitri P. Bertsekas (2007)

Figure 21. Generic Approximate Policy Iteration Block Diagram

5.4 Approximate policy iteration Based on Monte Carlo Simulation

As mentioned before because of the size of the state space, the number of possible values for the state variable is far too large to apply the policy iteration method directly. Overcoming this curse of dimensionality requires the introduction of an approximation architecture for the value function. In the following we present a version of approximate policy iteration based on Monte Carlo simulation, function approximation and the Least Squares regression model. The idea behind the following method was initially proposed by Bertsekas and Tsitsiklis (1996). Figure 22 illustrates a block diagram of approximate

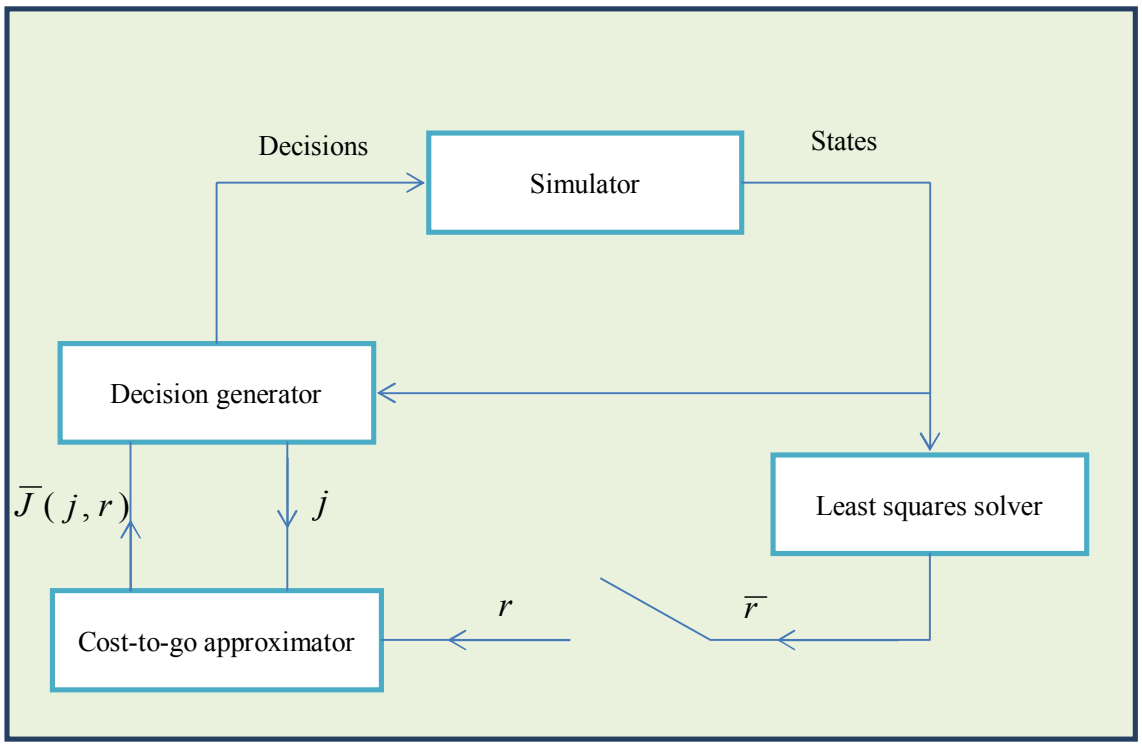
`

policy iteration based on the Monte Carlo simulation and demonstrates how this method fits within the approximate policy iteration framework.

The corresponding loop consisting of "Simulator" block and "decision generator" block can be regarded as the policy evaluation loop. The "Decision generator", generates the optimal action u based on the current policy to be used in the simulator. The "simulator" generates the next state j according to the transition probabilities in the current state i and the action u obtained from the decision generator. Once the policy has been fully evaluated in the policy evaluation loop and sample trajectories have been collected, then the "Least squares solver" is invoked to determine the optimal value of the approximation parameters $\bar{r}$ .The $\bar{r}$ is then used in the "Cost-to-go approximator" to replace the current value function approximation $\tilde{J}(j,r)$ by $\tilde{J}(j,\bar{r})$ to obtain the improved policy.

There are two potential sources of error however in these types of methods. One is the simulation noise generated by using the simulation experiments to tune the parameter vector r. The other is in choice of approximation architecture that may not be flexible enough to estimate the value function accurately enough.

`



Source: "Dynamic Programming and Optimal Control", 3rd Edition, Volume II, by Dimitri P. Bertsekas ((2007)

Figure 22. Approximate Policy Iteration Based on the Monte Carlo Simulation Block Diagram

5.5 Approximate Policy Iteration Implementation

In this section we present the detailed implementation of the simulation-based approximate policy iteration algorithm based on the Monte Carlo simulation that is used as our main solution methodology for tuning the approximation parameters in this thesis. We tune the parameters through an iterative method. Each iteration consists of two steps. In the first step the cost trajectory of the system is collected by simulating the long enough trajectory of the stationary policy μ driven from the current value function

`

approximation. In particular we generate an estimate of the cost function $C^n(q)$ using simulation for each post-decision state i' generated in replication q . We do this for Q different replications. In each replication we generate an initial random post-decision state i'. In the second step the parameters are tuned by solving the least-squares problem for the set of Q post-decision states and cost estimates. The regression model fits the value function approximation $J\left(i'^n(q), r\right)$ to the collected cost estimates $C^n(q)$. This works because the value function is the total discounted cost over the infinite horizon. The new set of parameters characterizes the new value function approximation that is used in the next repetition of the above steps. This iterative process is repeated until the tunable parameters converge to some stabilized value. The idea is closely similar to the classical policy iteration where the first step resembles the policy evaluation and the second step accounts for policy improvement. We start with an arbitrary parameter vector $\vec{r}^1$ and generate sequence r_t using the following procedure:

- Step 0.
  - Initialize the iteration counter n to 1.
  - Initialize tunable parameter vector $\vec{r}^1$ arbitrarily.
- Step 1. (Policy evaluation through simulation)
  - Step 1a. Set replication counter $q = 1$.
  - Step 1b. Initialization

`

- Generate post-decision state $i'^{n}(q)$
- Find the next pre-decision state associated with $i'^{n}(q)$

– Step1c. Do for $t = 1,...,T$:

- Generate an optimal control u_t by letting :

$$u_t = \arg\min_{u \in U} \left\{ c(i_t, u) + \gamma \tilde{J}\left(f_u\left(i_t, u\right), r_t \right) \right\}$$

- Compute cost associated with action u_t taken in current pre-decision state

$$C_t\left(i_t, u_t\right)$$

- Obtain the next post-decision state using control u_t :

$$i'_t = f_u\left(i_t, u_t\right)$$

- Run simulator to obtain the next pre-decision state using current post decision

$$i_{t+1} = f_w\left(i'_t, w_t\right)$$

– Step1d. Compute discounted cost incurred by starting from state $i'^{n}(q)$

$$C^n\left(q\right) = \sum_{t=1}^{T} \gamma^{(t-1)} C_t$$

– Step 1e. If $q \le Q$, increment q and go to Step 1a.

- Step 2. (Policy improvement)
  - Step 2a. (Projection) Compute the tunable parameters at the next iteration by solving the following lest square regression model :

`

$$\vec{r}^{*} = \arg\min_{\vec{r}} \left\{ \sum_{q=1}^{Q} \left[C^{n}(q) - J\left(i'^{n}(q), r\right) \right]^{2} \right\}$$

– Step2b. Update the value of the approximation parameters using step size α_n

:

$$\vec{r}^{\,n} = (1-\alpha_n)\vec{r}^{\,n-1} + \alpha_n \vec{r}^{*}$$

- Step 3. If $n < N$, increment n and go to Step 1.

The question remains as to how to generate the post-decision states at the start of each replication? The answer to this question is crucial as the least squares step is based on these initial post-decision states. The best approach is to simulate a policy for several days (warm-up period), starting from a random initial state. The initial random state could be generated using a uniform distribution. The policy that should be used during the warm-up period is the best policy that is normally known for the problem. But in our problem because we don't have a good initial policy, for generating initial post decision states in each run, we use a distribution that heavily weights "good states" and states that are more likely to be realistic. So for instance, we don't want to choose an initial state that is empty but neither do we want to choose one that is absolutely full. This ensures that we are tuning the parameters based on states that we are likely to visit in a realistic version of the problem. For the number of patients already booked, the underlying distribution chooses from a set of states where days are 1/2 to 9/10 full with higher

`

likelihood of being closer to 9/10 the closer to the day of surgery. For the number of patients in the waiting list we use the distribution of arrivals.

The discount factor γ plays an important computational role here in terms of reducing error. In particular, the linear value function approximation includes some error and the discount factor is used to place more emphasis on the present cost rather than future costs predicted by linear value function approximation. We chose 0.99 as a discount factor in our algorithm.

In step 2 of the algorithm we are dealing with a system of linear equations in the unknown parameter vector r .The least squares algorithm is very advantageous in that it does not require the scaling of features and also does not dependent on the initial estimate of weights (Bradtke and Barto 1996). But the drawback is that it is computationally expensive as it requires solving a matrix inverse for every iteration. This can be solved by an incremental gradient method and other methods of that type (Bertsekas and Tsitsiklis 1996) or it can be simply solved by applying methods available in linear algebra packages. We used the LAPACK linear algebra package in the Netlib repository to solve the least squares regression problem in Step 2; see Anon. (2004).

Coding of the algorithm has been done in C++ and all computations were performed on an Intel(R) Core(TM) 2 CPU with 1.6 GHz. We use IBM ILOG CPLEX to solve the linear optimization problem in step 1c of the algorithm; see Anon.(2012). In order to generate the necessary random numbers from a specific distribution especially in finding the next pre-decision function, the GNU Scientific Library (GSL) has been used. See

`

Anon. (2011) .The GNU Scientific Library (GSL) is a numerical library for C and C++ programmers.

The replication parameter Q has been set to 50 and the simulation horizon to $T = 100$ within each replication. The algorithm has been run for 300 iterations. The CPU time for each iteration of our ADP algorithm was almost 21 minutes including solving the linear optimization problem 5000 times and the enumeration over possible post-decision states. Once we have a good value function approximation, it takes less than three seconds to solve the least squares regression problem using LAPACK to obtain the new approximation parameters.

5.6 Step Size

An appropriate step size or learning rate α_n has to be used to update the value of the approximation parameters in step 2b of our approximate policy iteration algorithm. For $\alpha_n = 1$ the current value of the approximation parameters would be replaced by the new estimate of tunable parameters obtained by solving the least squares model. This might work in the value iteration method, as in value iteration the expectation has been taken over all possible states, but this is not the case in approximate dynamic programming because at each iteration the sample trajectory of only one post decision state is simulated leading to its sample estimate. In order to enable the algorithm to converge, the chosen step size should meet a few conditions. If the chosen step size is too large, it makes the

`

algorithm unstable, and if it is too small it forces the algorithm to converge before it has had time to best represent the value function.

A large number of rules for determining the best form of the step size have been proposed by Bertsekas and Tsitsiklis (1996) and Powell (2010). In our algorithm we use the diminishing step size $\alpha_n = \frac{a}{a+n-1}$ as proposed by Powell (2010) where n is the iteration number and a is a integer number. This form is called the "generalized harmonic step-size" which is the generalization of the $\frac{1}{n}$ rule where n is the iteration number.

5.7 Summary

In this chapter the ADP methodology has been applied for the purpose of solving the MDP model. Among different methods within ADP, the simulation based approximate policy iteration has been chosen as main solution methodology for tuning the approximation parameters. The appropriate algorithm is then developed to implement the method. The next chapter presents the results obtained from the implemented method and shows how the derived results could determine the policy for the surgical scheduling model.

`

Chapter 6: Result, Conclusion and Future Research

In this chapter we present the results obtained from the implemented ADP algorithm. Section 6.1 presents the performance of the ADP method in terms of the convergence behaviour of the approximation parameters. In Section 6.2, the derived policy is analyzed from the booking prospective. Section 6.3 presents the results of simulating the derived ADP policy and comparing it with a First Come First Served (FIFO) policy in terms of the performance measures. In Section 6.4 the conclusion and contribution of the thesis is discussed and at the end the future extensions of this work are outlined.

6.1 Convergence Behavior

In this section the convergence results of the approximation parameters is investigated. As mentioned in chapter 5, the $\alpha_n = \frac{a}{a+n-1}$ step size rule has been applied in the implemented ADP algorithm. The algorithm has been tested for $a = 1, 3, 5$ and 20 with almost identical results for estimated parameters. It is not surprising that the approximate policy iteration converges a lot faster compared to value iteration or other

`

methods of that type. That is primarily because the algorithm is fully evaluating the current policy at each iteration to find a good approximation of the value function $J^{\mu}(i,r)$. Convergence results for some tunable parameters are reported in the following graphs. In the following, the convergence behavior of approximation parameters is presented for the urology and thoracic services as for instance. The results don't differ for other surgical services included in the model. Figure 23 illustrates the behavior of the r_0 parameter of the approximation architecture (Eq. (5.11)) which has begun to stabilize after 100 iterations.

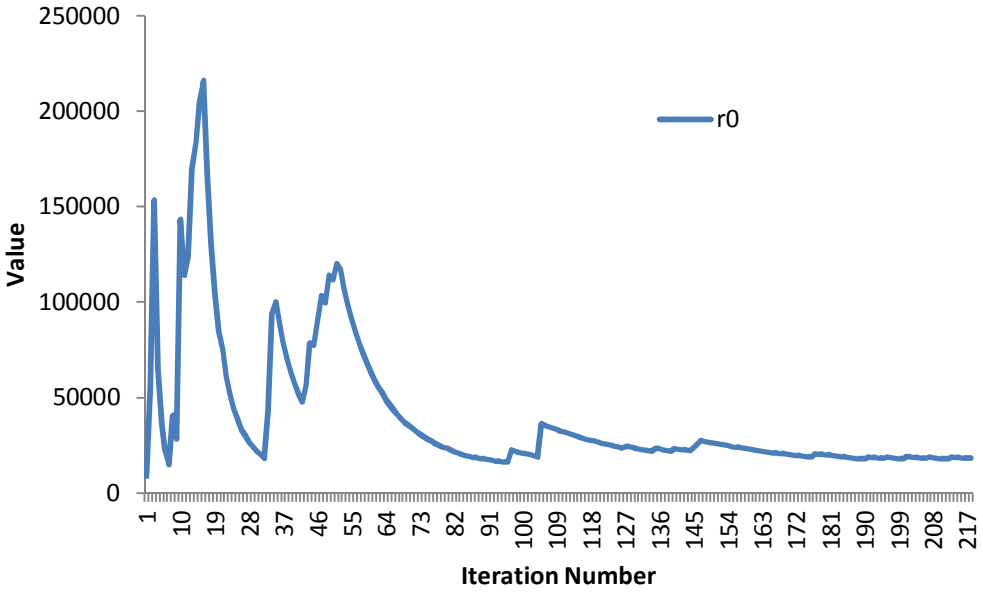


Figure 23. Convergence Behaviour of Parameter r_0

The following two graphs present the behavior of the r_i^w and r_i^y approximation parameters for class i patients within the urology service.

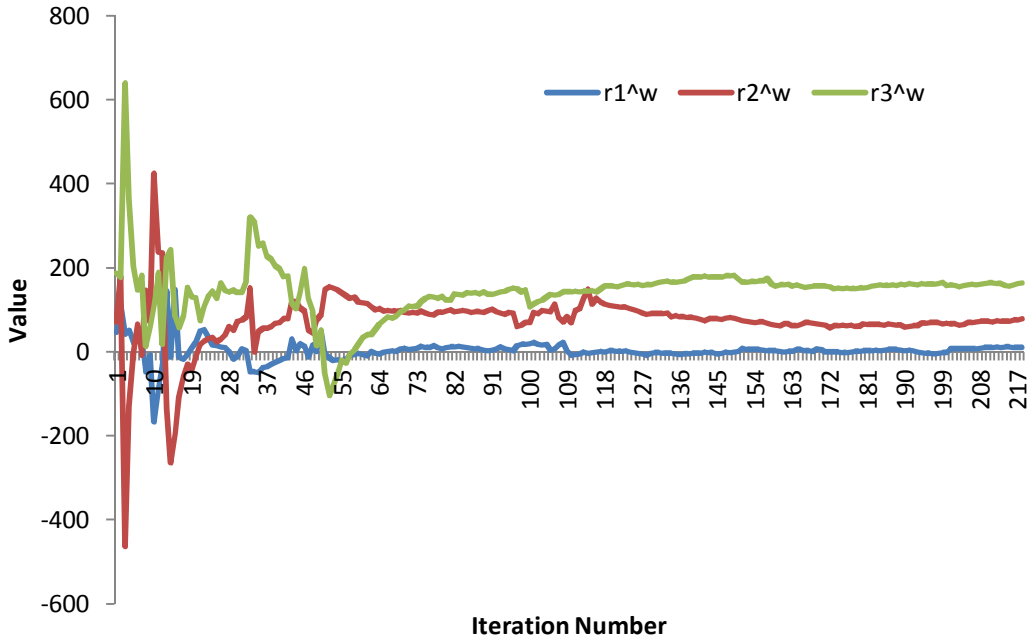


Figure 24. Convergence Behaviour of r_w Parameter, Urology Service

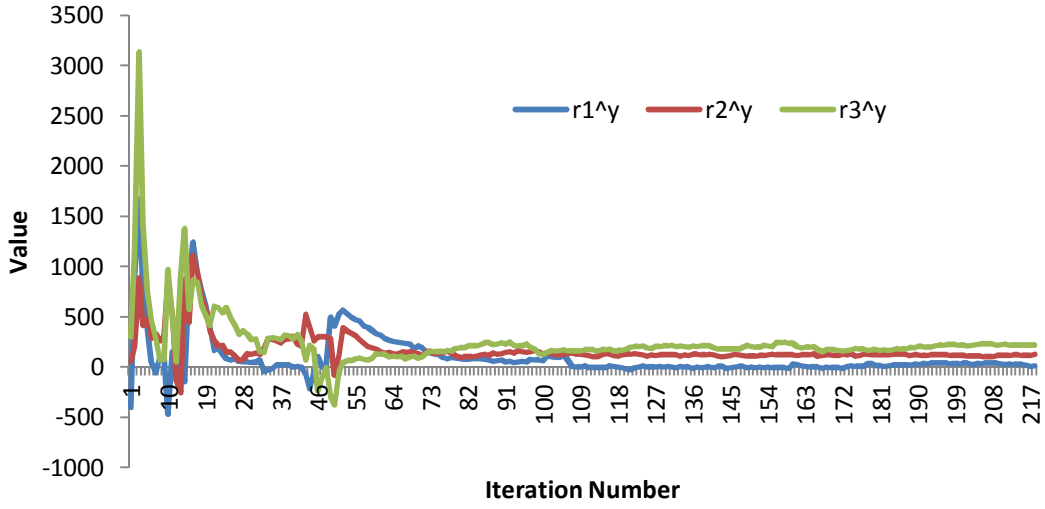


Figure 25. Convergence Behaviour of r_y Parameter, Urology Service

`

The following two graphs present the behavior of r_i^w and r_i^y parameters for class i patients within the thoracic service.

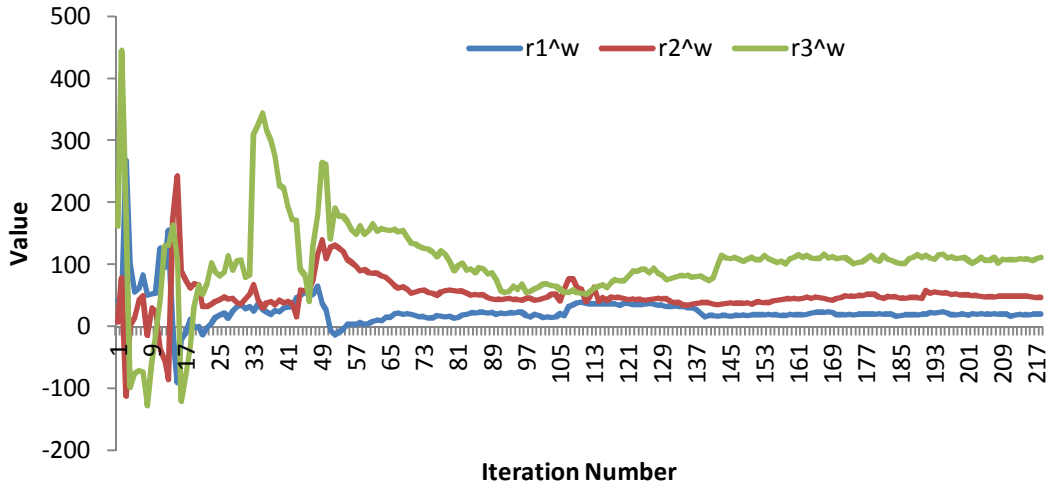


Figure 26. Convergence Behaviour of r_w Parameter, Thoracic Service

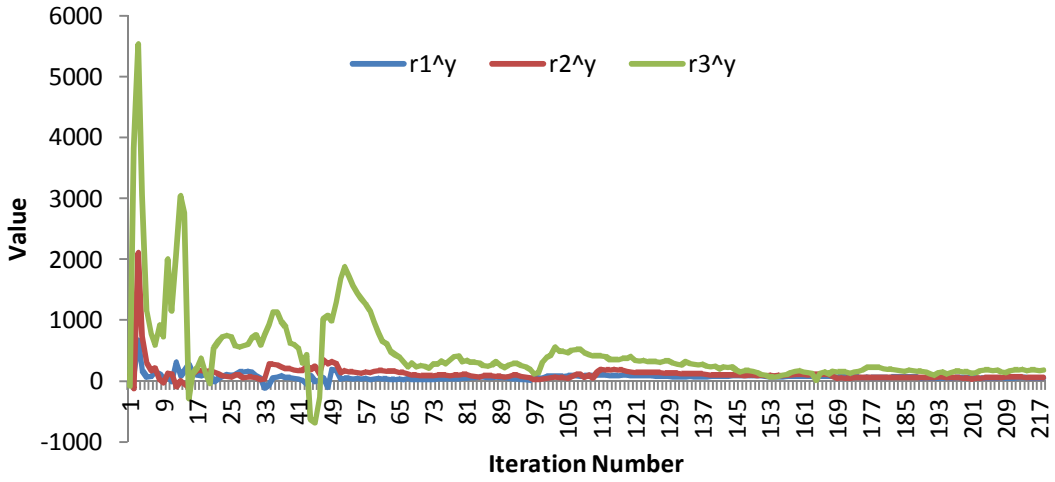


Figure 27. Convergence Behaviour of r_y Parameter, Thoracic Service

`

In the graphs above, there is a "learning period" where the algorithm is adjusting the parameter values. All the approximation parameters have begun to stabilize after that learning period is over. The final values of these parameters after convergence will form the shape of the value function approximator.

According to the convergence graphs, within each surgical specialty service, the value of r^w and r^y parameters converge to higher values as the class number increases. The reason is that on one hand within each service, patients in higher classes consume more of resources because of longer length of stay in hospital and surgery length and therefore the marginal cost of booking a patient in higher class result in higher value of r^w. On the other hand, in the MDP model formulation, the higher priority is given to classes which have lengthier surgeries and longer length of stay within each surgical service. These naturally represent the classes containing more complex surgeries and thus higher priority surgeries. Therefore the marginal cost of keeping patients on the waitlist is higher for higher classes due to the result of our prioritization, so intuitively their associated parameter should converge to a higher value.

6.2 ADP Policy Analysis

As the ADP algorithm progressed, it began to realize that high levels of overtime eventually lead to high costs and adjusted the parameter estimates accordingly. Once the

`

learning period is over then final parameter values provide a trade-off between delaying patients versus using overtime excessively.

We can investigate the derived policy from the booking perspective by analyzing the Bellman optimality equation used for taking actions. By expanding the Bellman optimality equation, we get the following action cost:

$$\sum_{i\in I} f^{Delay}(i)\left(y_i - \sum_{n\in N} b_{in}\right) + \sum_{(i,n)\in I\times N} f^{Late}(i,n)b_{in} + \gamma\left(r_0 + \sum_{(i,n)\in I\times N}\left(w_{in}+b_{in}\right)r_i^w + \sum_{i\in I}\left(y_i - \sum_{n\in N} b_{in}\right)r_i^y\right)$$

By expanding the above equation even more we have:

$$\sum_{i\in I} f^{Delay}(i)y_i - \sum_{(i,n)\in I\times N} f^{Delay}(i)b_{in} + \sum_{(i,n)\in I\times N} f^{Late}(i,n)b_{in} + \gamma r_0 + \gamma\sum_{(i,n)\in I\times N} w_{in}r_i^w + \gamma\sum_{(i,n)\in I\times N} b_{in}r_i^w + \gamma\sum_{i\in I} y_i r_i^y - \gamma\sum_{(i,n)\in I\times N} b_{in}r_i^y$$

Ignoring the constant terms we get:

$$-\gamma\sum_{(i,n)\in I\times N} f^{Delay}(i)b_{in} + \sum_{(i,n)\in I\times N} f^{Late}(i,n)b_{in} + \gamma\sum_{(i,n)\in I\times N} b_{in}r_i^w - \gamma\sum_{(i,n)\in I\times N} b_{in}r_i^y$$

In the equation above $\gamma\sum_{(i,n)\in I\times N} f^{Delay}(i)b_{in} + \gamma\sum_{(i,n)\in I\times N} b_{in}r_i^y$ represents the "delay cost" and $\gamma\sum_{(i,n)\in I\times N} b_{in}r_i^w + \sum_{(i,n)\in I\times N} f^{Late}(i,n)b_{in}$ represents the "booking cots". By substituting the estimated parameter values derived from the ADP algorithm and the

`

value for f^{Delay} similar to what has been used during the implementation of ADP, and considering the fact that the late cost is only effective after the wait time target is passed, we can see:

$$r_i^w < f^{Delay}(i) + r_i^y$$

which means that the booking cost for patient class i is less that the delay cost. This in turn implies that the ADP policy prefers to overbook surgeries rather than postponing the booking decision. This is true for all patient classes in the model.

It also can be seen that the difference between the delay cost and the booking cost $f^{Delay}(i) + r_i^y - r_i^w$ is higher for higher classes within each service which means that the policy gives booking priority to higher classes. This is also true for all surgical services in models.

6.3 Simulation Results

Because the approximate policy was derived through ADP, there is no guarantee of its optimality. Therefore the behaviour of the resulting policy should be investigated through simulation and possibly compared to an existing policy. This section presents the results of such a simulation. The performance metrics of interest collected during the simulation are:

`

1. The OR utilization for each surgical specialty on each day of booking horizon measured by: $z_{jn} - \sum_{i \in I(j)} \left(b_{in} + w_{in} \right) \qquad \forall j, n \in [J] \times [N]$

2. Waiting demand as the number of patients from class i patients waiting to be booked every day measured by: $\left(y_i - \sum_{n \in [N]_0} b_{in} \right)$

3. Post-operative bed utilization measured by: $\sum_{(i,n) \in [I] \times [M(i)]} x_{in}$

The simulation of the scheduling process has been done in C++ using IBM ILOG CPLEX as a solver as it involves solving the integer program given in step 1c of the ADP algorithm subject to the constraints (3.18) to (3.21). We ran the simulation model for 1000 days with a warm-up period of 100 days. At this stage each surgical service can be simulated separately to investigate its dedicated OR utilization and waiting demand trends as each service functions independently in terms of OR time and waiting demands, though they all use the same pool of post-operative beds. Therefore, for the bed utilization, multiple services are involved in the simulation model. Therefore, in the following the test results for the orthopedics surgical service are presented which according to our data has more demand than allocated capacity. The results are similar for other surgical services and the derived policy shows similar behavior provided they have capacity constraints. The ADP policy is tested against the First Come First Served (FIFO) policy and the performance measures are compared with each other. The FIFO policy manages to book the demands as soon as they arrive, as long as the surgical time

`

fits into the available block regardless of priority. Once the demand arrives, the policy books it into the earliest available block only if the surgical time fits; otherwise the demand is postponed to the next available block. This is our best attempt to mimic the current practice at the Ottawa Hospital. In our simulation model, the surgical time and length of stay for each patient is generated from the specific distribution determined from the data for each patient class. The demand is also generated according to the parameters that have been estimated for the arrival rate of each class. As discussed in Chapter 4, for most patient classes, the arrival rate follows a Poisson distribution. The available capacity has been set according to the regular block times in the master schedule available in our data set with the booking horizon running for 20 days. Table 1 shows some of the parameters used in the simulation model including the distributions of surgical times and arrival rates for each patient class within orthopedics service.

Simulation Parameters	Surgical times	Arrival rates
Class 1	Lognormal ($\sigma = 0.67, \mu = 4.2$)	Poisson(4.95)
Class 2	Lognormal ($\sigma = 0.49, \mu = 4.8$)	Poisson(3.60)
Class 3	Lognormal ($\sigma = 76, \mu = 5.3$)	Poisson(0.24)

Table 1. Simulation Parameters for Orthopedics Surgical Service

Table 2 shows probability of discharge estimated for the patient's length of stay in hospital for each class of the orthopedic surgical service. In particular this table shows the probability of a patient of class i being discharged after n days. As mentioned before,

`

this probability graph stabilizes after a period of time so we can simply track that point as the maximum length of stay and assign a probability to for the category of patients who stay longer than that. For class 1 in the table for instance, the value of 0.1 has been assigned for the category of patients who stay longer than 15 days.

	Class 1	Class 2	Class 3		Class 1	Class 2	Class 3
Day 0	0.27	0.03	0.00	**Day 11**	0.10	0.09	0.00
Day 1	0.34	0.22	0.16	**Day 12**	0.08	0.03	0.09
Day 2	0.10	0.15	0.13	**Day 13**	0.07	0.10	0.10
Day 3	0.20	0.19	0.09	**Day 14**	0.07	0.07	0.05
Day 4	0.28	0.22	0.14	**Day 15**	0.08	0.06	0.22
Day 5	0.26	0.19	0.25	**Day 16**	0.09	0.07	0.14
Day 6	0.20	0.15	0.20	**Day 17**	0.10	0.06	0.14
Day 7	0.17	0.14	0.03	**Day 18**	0.10	0.07	0.14
Day 8	0.16	0.16	0.00	**Day 19**	0.10	0.14	0.14
Day 10	0.11	0.08	0.12	**Day20**	0.10	0.11	0.14

Table 2. Discharge Probability for Each Day of Stay in Hospital for Orthopedics Service

The following three sections provide the comparative results of the simulation model for the ADP and FIFO policies. All data set and simulation parameters used for both scenarios are similar. The same seed has been used for generation of the random numbers in both scenarios to ensure valid comparison between the policies.

`

6.3.1 Waiting Demand

In this section, the behaviour of the ADP and FIFO policies are investigated in terms of the waiting demands. Figure 28 and 29 illustrate the time series graph of the waiting demand for all patient classes involved in the orthopedics surgical service. According to Figure 28, under the ADP policy, the number of patients waiting to be booked from class 3 levels out after an initial increase and remains fairly stable throughout the simulation horizon. Class 2 rarely has any waiting demand while class 1 has quite high fluctuations in terms of the number waiting. In contrast to the ADP policy, for the FIFO policy (see Figure 29) there is no waiting demand from class 1, high fluctuations in the number of patients waiting from class 2 and most significantly a steadily increasing wait list for patients from class 3. In the ADP policy the highest priority is given to class 3 which have lengthier surgeries and longer LOS. These naturally represent the classes containing more complex surgeries and thus higher priority surgeries. Thus, our policy outperforms the FIFO policy in terms of better managing waiting demand when there is limited capacity and there exists different patient classes with different priorities. The reasoning is fairly intuitive. The FIFO policy manages to fit the shorter surgeries in quite easily but often has difficulty finding enough capacity for the longer surgeries. The ADP policy, on the other hand, makes sure that sufficient capacity is found for the longer surgeries and then fits the shorter surgeries in around the longer ones. This leads to a stable waiting

`

lists for the longer surgeries (class 3) and a fairly volatile one for the shorter surgeries (class 1). It also makes for a much more consistent use of available capacity.

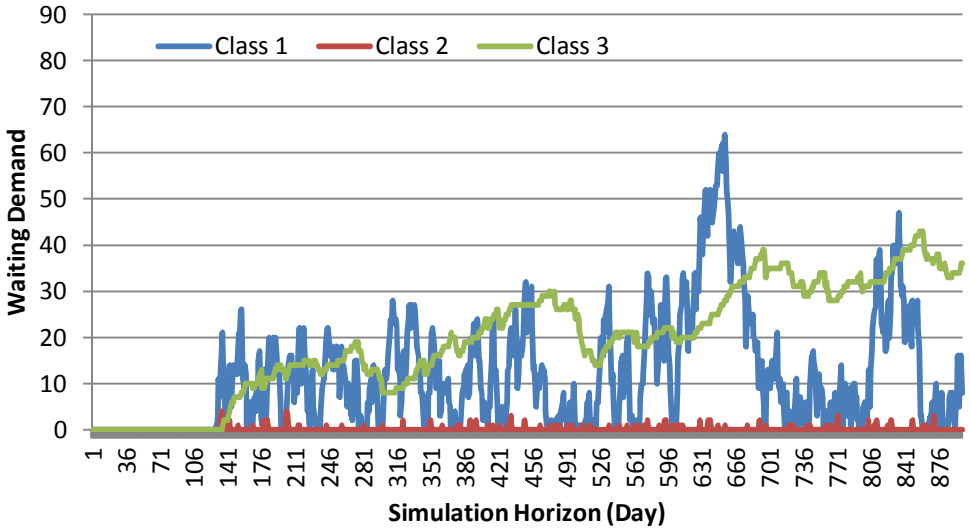


Figure 28. Waiting Demand Time Series for the ADP Policy

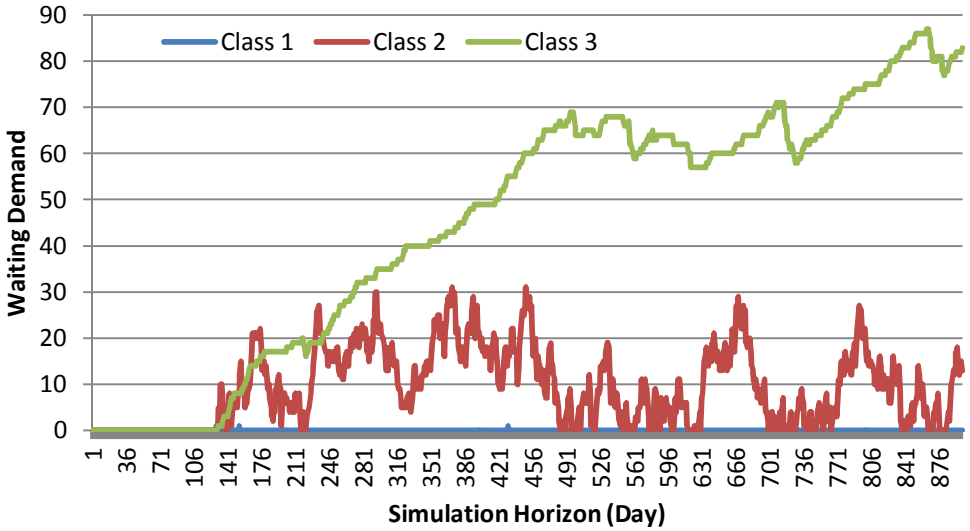


Figure 29. Waiting Demand Time Series for the FIFO Policy

`

6.3.2 Operation Room Utilization

In this sub-section, behaviour of both policies is compared in terms of the utilization of the operating rooms (ORs) throughout the booking horizon as well as the number of patients being serviced by each policy throughout the entire simulation. Regular surgical block times are used without any added overbooking capacity meaning that the sum of the *expected* surgical times booked into a block has to be less than the total block time. Figure 30 provides the sum of the expected surgical times booked for surgery plus given setup and teardown times (provided by TOH) between surgeries from day 1 to day 20 of the booking slate divided by the available capacity. Both policies tend to book at least 10 days out as evidence by the over 90% utilization by day 10. For the day of surgery, represented in Figure 30 as day 0, actual surgery times for each of the booked patients has been generated according to the lognormal distribution estimated for each patient class. The sum of these surgical times added to given set-up and teardown times between surgeries represents the estimate of total surgical time on the day of surgery. It might seem surprising that despite the restriction of the sum of the expected surgical times being less than the OR capacity, day zero still results in overtime. The reason is that we restricted the sum of the expected surgeries to be less than OR capacity rather than the expectation of the sum. Table 3 compares utilization percentages from figure 30. According to the table in ADP policy scenario, ORs have 21% more utilization in total (equal to 1.04% per day on average) than FIFO policy indicating significantly greater

`

advance planning. In addition, on the day of surgery, the ADP policy results in slightly higher rates of overtime utilization which is how it maintains a stable wait list.

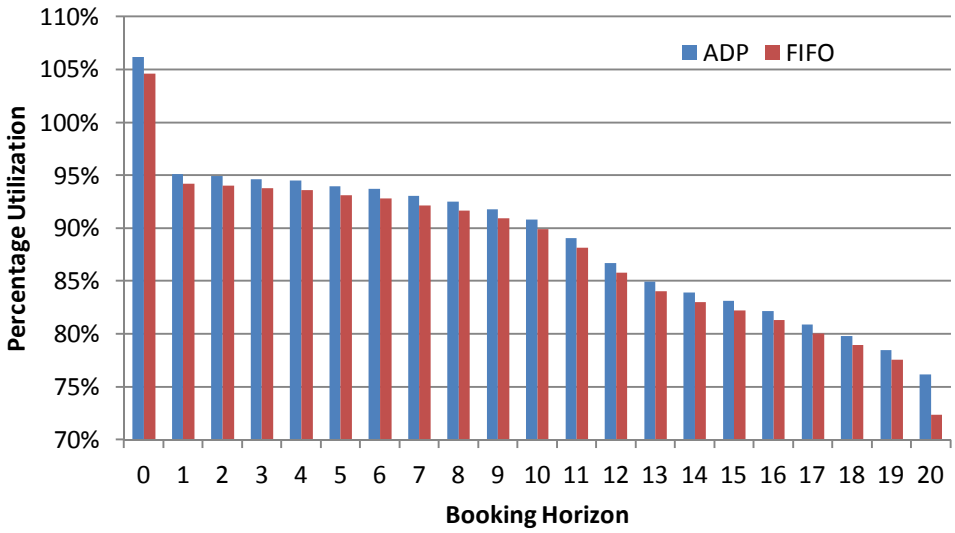


Figure 30. Average OR Utilization of Both Policies for Orthopedic Surgical Service

	ADP	FIFO	ADP-FIFO		ADP	FIFO	ADP-FIFO
Day 0	1.062	1.046	0.015	**Day 11**	0.891	0.882	0.009
Day 1	0.951	0.943	0.009	**Day 12**	0.867	0.858	0.009
Day 2	0.949	0.941	0.009	**Day 13**	0.849	0.841	0.009
Day 3	0.947	0.938	0.009	**Day 14**	0.839	0.831	0.009
Day 4	0.945	0.936	0.009	**Day 15**	0.831	0.823	0.009
Day 5	0.940	0.931	0.009	**Day 16**	0.822	0.813	0.009
Day 6	0.937	0.929	0.009	**Day 17**	0.809	0.800	0.009
Day 7	0.931	0.922	0.009	**Day 18**	0.798	0.789	0.009
Day 8	0.925	0.917	0.009	**Day 19**	0.785	0.776	0.009
Day 9	0.918	0.910	0.009	**Day 20**	0.762	0.724	0.038

`

Day 10	0.908	0.899	0.009	**Total**	18.667	18.444	**0.219**

Table 3. OR Utilization (Percentage)

Because the simulation model has been run under similar conditions for both scenarios and due to the fact that the same seed has been used in these scenarios, we can confidently compare the total number of patients being served by each policy. Table 4 present such a comparison. According to the table, the total number of patients served by the ADP policy is greater than the FIFO policy though admittedly the difference is not huge. In addition to this, the table clearly shows that the ADP policy is serving more patients from class 3 which indicates better management of priorities compared to the FIFO policy.

	Class 1	**Class 2**	**Class 3**	**Total Served**
ADP	54.45 %	43.34 %	2.21 %	8895
FIFO	54.46 %	44.06 %	1.49 %	8886

Table 4. Number of Patients Served by Both Policies

6.3.3 Bed Utilization

Figure 31 shows the histogram of the bed utilization for both policies. As stated before, the ADP model was solved for a system involving all available services and the resulting approximation is used in the simulation. In contrast to the OR utilization and

`

waiting demand, for the simulation of bed utilization, multiple services are involved because all surgical services use the same pool of post-operative beds. In particular, orthopedics and urology surgical services are considered in this scenario. Setting the number of available beds to 75, table 5 shows that the ADP policy has slightly higher levels of the utilization, but no major difference exists when it comes to comparing the histogram of both policies. It must be remembered that the ADP policy is also managing to serve a higher number of high class patients in the process who have much longer lengths of stay while maintaining similar utilization rates on the ward. Table 5 also shows the total throughput for both policies which represents total number of the patients in the ward during the simulation horizon. This number is clearly greater for ADP policy meaning serving more patients while maintaining similar utilization rates on the ward.

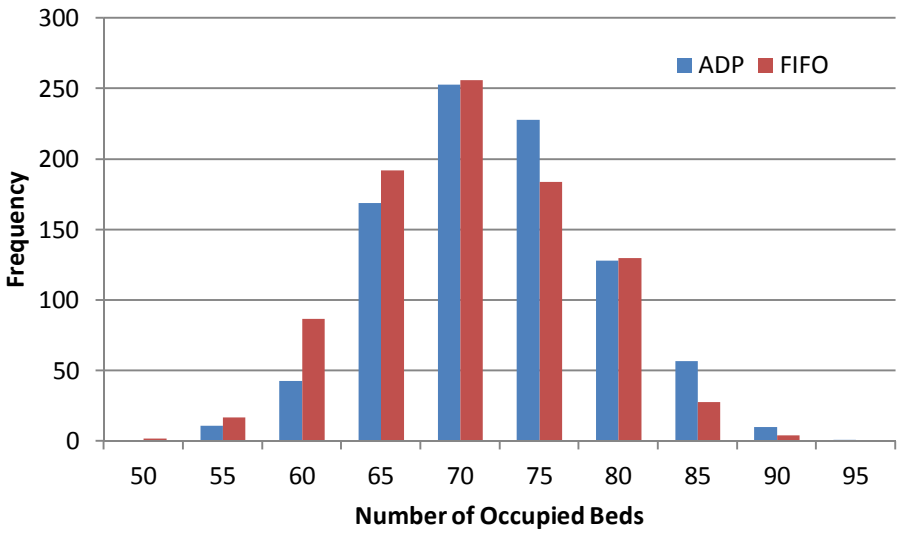


Figure 31. Bed Utilization Histogram for Both Policies

`

	Utilization Rate	Throughput
ADP	93.75 %	63277
FIFO	91.61 %	61842

Table 5. Bed Utilization Rate for Both Policies

6.4 Conclusions and Contributions

In this thesis, we developed a comprehensive Markov Decision Process model for the surgical scheduling problem. The results from the simulation indicate that the ADP policy outperforms the FIFO policy in terms of better wait list management in cases where capacity is an issue.

Among the methods of ADP, the Approximate Policy Iteration method provided the best results for our surgical scheduling problem. In this research we applied some variants of Temporal Difference (TD) learning methods in addition to Approximate Policy Iteration, but with less satisfactory results.

In contrast to other approaches to the surgical scheduling problem such as those based on integer programming, our MDP methodology is more beneficial as it captures the stochastic evolution of the system over time. Moreover, the decision can be made very quickly by our method as it only requires solving a simple linear optimization problem that minimizes the sum of the immediate cost and the value function approximation. In other approaches, one is usually limited to solving a small version of the model due to the

`

computational difficulties; whereas in the ADP approach the larger models can be solved with realistic dimensions. On the other hand, due to the approximation error, this approach will never give us the optimal solution to the problem.

6.5 Future Extensions

In this section a few possible extensions of this work are enumerated:

First, in this thesis we have used a linear architecture to approximate the value function but it is quite possible that the value function might be better represented by non-linear approximation architecture. Thus one possible extension of the work could be trying to approximate the value function with a nonlinear approximation architecture such as those based on the logistic curves and investigate the quality of the derived policy.

In this research, the simplified version of the model has been addressed and the solution methodology has been implemented on a small sized problem. Now that the methodology has been demonstrated for the simplified version, one can begin to include additional complexity. Therefore, another possible extension of this work is to try to solve the comprehensive MDP model developed in Chapter 3.

In this thesis we have used simulation based approximate policy iteration methodology from the domain of simulation based ADP methods to solve the MDP model. As stated before there are several other methods in this category. We had an unsuccessful attempt to employ Temporal difference (TD) methods for this problem. One

`

might try to develop an appropriate algorithm based on the temporal difference or apply other alternative methods from this category.

Another possible extension of the work could be to allow the decision maker to dynamically reserve bed capacity a fixed number of days in advance in the full MDP model presented in chapter 3.

Lastly, in this research we the focus was on the third stage of general elective surgery scheduling - determining the number of patients to be scheduled into each available block. Two potential extensions are incorporating the determination of the optimal master schedule into the model and incorporating an algorithm for determining the optimal appointment time for each surgery on the day of surgery in order to get better estimates of overtime.

`

Appendix A: Data Collection

This appendix provides the summery of data collection provided to us by the Ottawa hospital. Data collection can be divided into the following parts:

Part 1: WTIS

Verbal description of cohort	All surgical requests in WTIS with the decision to treat date from January through December 2010, including the open cases as of the end of Dec 2010.
Start/End Dates	Jan 01, 2010 – Dec 31, 2010

Definition of data table

	Column label	**Derivation description**
1.	WTIS Record ID	De-identified unique record ID in WTIS
2.	Patient ID	De-identified patient ID
3.	Procedure	One patient might have more than one surgical procedure scheduled in WTIS
4.	Site	e.g. Campus

`

5.	Physician ID	De-identified physician ID derived from health care professional
6.	Priority	
7.	Status	Open/Closed
8.	Access Target (Days)	
9.	Dates Affecting Readiness to Treat (Days)	
10.	Total Wait (Days)	
11.	Variance (Days)	
12.	Decision to Treat Date	
13.	Scheduled Procedure Date	
14.	Actual Procedure Date	
15.	Procedure No Longer Required Reason	

Part 2. SIMS

Verbal description of cohort	All OR visits in SIMS with surgery date from January through December 2010
Start/End Dates	Jan 01, 2010 – Dec 31, 2010

Definition of data table

	Column label	**Derivation description**
1.	SIMS Record ID	De-identified unique record ID in SIMS
2.	Encounter ID	De-identified encounter ID
3.	Patient ID	De-identified patient ID

`

4.	Physician ID	De-identified physician ID derived from health care professional
5.	OR Visit Sequence Number	Sequence Number per visit to OR
6.	Multiple Procedure Flag	Flag whether or not the patient has multiple procedures per visit to OR
7.	Type of Surgery	Elective vs. Emergency
8.	Principle Procedure	
9.	Site	e.g. Campus
10	Eligible for WTIS	Flag whether or not the procedure is eligible for SIMS.
11	Priority	
12	Surgical procedure start datetime	
13	Surgical procedure end datetime	
14	Length Of Surgical Procedure	In minutes

Part 3. DW

Verbal description of cohort	All linked OR visits from SIMS by encounter numbers
Start/End Dates	Jan 01, 2010 – Dec 31, 2010

Definition of data table

	Column label	**Derivation description**	**Data Source, if the variable is from DW**
1.	SIMS Record ID	De-identified unique record ID in SIMS	SIMS
2.	Patient ID	De-identified patient ID	SIMS, Encounter table in DW

`

3.	Encounter ID	De-identified encounter ID	SIMS, Encounter table in DW
4.	OR Visit Sequence Number	Sequence Number per visit to OR	Derived from SIMS
5.	Post-operative length of Stay in ICU per case in DW	In hours, to end of next case or till discharge. Note: for multiple procedures within the same case, their "Length of Stay in ICU" will be same and only available for Inpatient.	Inpatient Census History table
6.	Post-operative length of Stay in Acute Wards (excluding ICU) per case in DW	In hours, to end of next case or till discharge.	Inpatient Census History table
7.	Most Responsible Diagnosis	Only available for inpatient and ED patient	DAD and NACRS
8.	Most Responsible Diagnosis Description	Only will be available for inpatient and ED patient	Derived
9.	Entry	Inpatient via ED, Inpatient Direct, and Daycare, only available for inpatient	DAD
10.	Encounter start datetime	Start datetime of encounter	Encounter table in DW
11.	Encounter end datetime	End datetime of encounter	Encounter table in DW
12.	Encounter type	Encounter type (i.e. inpatient, outpatient)	Encounter table in DW

Part 4. Master Schedule Template

`

ROOM	WEEK 1					WEEK 2				
	Monday	Tuesday	Wednesday	Thursday	Friday	Monday	Tuesday	Wednesday	Thursday	Friday
LATE ROOM COUNT	2	2	2	2	1	2	2	2	2	1
1- GENL Integrated	**Gen**	**Gen (L)**	**Gen**	**Gen**	**Gen**	**Gen**	**Gen (L)**	**Gen**	**Gen**	**Gen (L)**
	2	2	2	2	2	2	2	2	2	2
	Moonje	Auer	Arnaout	Tadros	Friedlich	Lorimer	Auer	Pitt	Tadros	Friedlich
2 - OPHTH		**Ophth**	**Ophth**	**Ophth**	**Ophth**			**Ophth**	**Ophth**	**Ophth**
		0	0	0	0			0	0	0
		Shared Ophth	Shared Ophth	Leonard	Patel			Shared Ophth	Leonard	Patel
3- GENL Integrated	**Gen (L)**	**Urol (L)**		**Gen**	**Gen**	**Gen (L)**		**Gen**	**Gen**	
	2	2		2	2	2		2	2	
	Boushey	Cagiannos		Lorimer	Arnaout	Boushey		Auer	Friedlich	
4 - Genl	**ACS** 8-12	**ACS** 8-12	**ACS** 8-12	**ACS** 8-12	**ACS** 8-12	**ACS** 8-12	**ACS** 8-12	**ACS** 8-12	**ACS** 8-12	**ACS** 8-12
	Emerg-Non Ortho	Emerg-Non Ortho	Emerg-Non Ortho	Emerg-Non Ortho	Emerg-Non Ortho	Emerg-Non Ortho	Emerg-Non Ortho	Emerg-Non Ortho	Emerg-Non Ortho	Emerg-Non Ortho
	0	0	0	0	0	0	0	0	0	0
5 - PLASTIC	**ENT PB**	**Plast**	**ENT**	**Plast**	**Plast**		**Plast**	**Gen**	**Plast**	
		1	2	0	1		1	2	1	
		Rockw ell	McLean	Jarmuske			Rockw ell	Arnaout	Jarmuske	
6 - UROLOGY	**Additional**	**Urol**	**Urol**	**LRT**	**Urol**		**Urol**	**Urol (L)**	**LRT**	**Urol**
	Oncology-Uro	2	2	2	2		2	2	2	2
	Breau	Warren	Mahoney		Cagiannos		Cagiannos	Breau		Cagiannos
7 - UROLOGY Integrated	**Urol**	**Urol**	**Urol**	**Urol**	**Urol**	**Urol**	**Urol**	**Urol**	**Urol**	**Urol**
	2	2	2	2	2	2	2	2	2	2
	Warren	Blew	Oake	Mahoney	Mahoney	Shared Urol	Oake	Watterson	Watterson	Mahoney
8 - ENT	**ENT (L)**	**ENT**	**ENT**	**ENT**	**ENT (L)**	**ENT (L)**	**ENT (L)**	**ENT**	**ENT**	**ENT**
	2	2	2	2	2	2	2	2	2	2
	Odell	Corsten	Lamothe	Odell	Lamothe	Odell	Corsten	Lamothe	Brow nrigg	Corsten

Figure 32. Portion of the Monthly Master Schedule at the Ottawa Hospital

`

Bibliography

Anon., 2004 August 21. *Linear Algebra PACKage.* [Online] Available at: http://www.netlib.org/lapack/

Anon., 2011 August 21. *GNU Scientific Library (GSL).* [Online] Available at: http://www.gnu.org/software/gsl/

Anon., 2012 August 21. *IBM ILOG CPLEX.* [Online] Available at: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

Bassamboo , A., Harrison, J. M. & Zeevi , A., 2006. Design and Control of a Large Call Center: Asymptotic Analysis of an LP-Based Method. *Operations Research,* 54(3), pp. 419-435.

Begen, M. A., 2010. *Appointment scheduling with discrete random durations and applications,* Vancouver: s.n.

Begen, M. A. & Queyranne, M., 2011. Appointment scheduling with discrete random durations. *Mathematics of Operations Research,* 36(2), pp. 240-257.

Bellman, R., 1954. The theory of dynamic programming. *Bull. Amer. Math. Soc.,* 60(6), pp. 503-515.

Bellman, R., 1957. *Dynamic Programming.* s.l.:Princeton University Press.

Bertsekas, D. P., 2007. *Dynamic Programming and Optimal Control.* s.l.:Athena Scientific.

Bertsekas, D. P. & Tsitsiklis, J., 1996. *Neuro-Dynamic Programming.* s.l.:Athena Scientific.

`

Blackwell, D., 1962. Discrete dynamic programming. *Annals of mathematical statistics,* Volume 33, pp. 719-726.

Blake, J. T. & Donald , J., 2002. Mount Sinai Hospital Uses Integer Programming to Allocate Operating Room Time. *Interfaces,* 32(2), pp. 63-73.

Bradtke, S. & Barto, A., 1996. Linear least-squares algorithms for temporal difference learning. *Machine Learning,* Volume 22, pp. 33-57.

Carter, M., 2002. Diagnosis: Mismanagement of resources. *OR/MS Today,* 29(2), pp. 26-32.

de Farias, D. & Van Roy, B., 2003. The linear programming approach to approximate dynamic programming.. *Operations Research,* 51(6), pp. 850-865.

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

Dexter , F., Traub , R. & Macario, A., 2003. How to release allocated operating room time to increase efficiency: predicting which surgical service will have the most underutilized operating room time. *Anesthesia & Analgesia,* 96(2), pp. 507-12.

Dexter, F. & Macario , A., 2004. When to release allocated operating room time to increase operating room efficiency. *Anesthesia & Analgesia,* 98(3), pp. 758-762.

Dexter, F. & Traub, R., 2002. How to schedule elective surgical cases into specific operating rooms to maximize the efficiency of use of operating room time. *Anesthesia & Analgesia,* 94(4), pp. 933-942.

Gerchak, Y., Diwakar , G. & Mordechai , H., 1996. Reservation Planning for Elective Surgery Under Uncertain Demand for Emergency Surgery. *Management Science,* 42(6), pp. 321-334.

Gosavi, A., 2003. *Simulation-based optimization: Parametric optimization techniques and reinforcement learning.* Boston(Dordrecht): Kluwer Academic Publishers.

Green, L. V., Savin, S. & Ben , W., 2006. Managing Patient Service in a Diagnostic Medical Facility. *Operations Research,* 54(1), pp. 11-25.

`

Gupta, D. & Wang , L., 2008. Revenue Management for a Primary-Care Clinic in the Presence of Patient Choice. *Operations Research,* 56(3), pp. 576-592.

Herring, W., 2011. *Prioritizing patients: Stochastic dynamic programming for surgery scheduling and mass casualty incident triage,,* s.l.: s.n.

Howard, R. . A., 1960. *Dynamic Programming and Markov Proces.* Cambridge(MA): MIT Press.

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,* Volume 185, pp. 1026-1037.

Macario, A., Vitez, T., Dunn, B. & McDonald, T., 1995. Where are the costs in perioperative care? Analysis of hospital costs and charges for inpatient surgical care.. *Anesthesiology,* 83(6), pp. 1138-1144.

Marbach, P., Mihatsch, O. & Tsitsiklis, J. N., 2000. Call admission control and routing in integrated services networks using neuro-dynamic programming. *IEEE Journal On Selected Areas In Communications,* 18(2), pp. 197-208.

McManus, M. et al., 2003. Variability in Surgical Caseload and Access to Intensive Care Services. *Anesthesiology,* 98(6), pp. 1491-1496.

Min, D. & Yih, Y., 2010. Scheduling elective surgery under uncertainty and downstream capacity constraints. *European Journal of Operational Research,* pp. 642-652.

Ogulata, S. & Erol, R., 2003. A hierarchical multiple criteria mathematical programming approach for scheduling general surgery operations in large hospitals. *Journal of Medical Systems,* 27(3), pp. 259-70.

Ozkarahan, I., 2000. Allocation of surgeries to operating rooms by goal programing. *Journal of Medical Systems,* 24(6), pp. 339-378.

Patrick, J., 2006. *Dynamic Patient Scheduling for a Diagnostic Resource,* Vancouver: s.n.

Patrick, J., L. Puterman, M. & Queyranne, M., 2008. Dynamic Multipriority Patient Scheduling for a Diagnostic Resource. *Operations Research,* 56(6), pp. 1507-1525.

`

Powell, W., 2010. *Approximate Dynamic Programming-Solving the curses of dimentionality-Second edition.* s.l.:John Wiley and Sons.

Powell, W. B., 2007. *Approximate dynamic programming - Solving the curses of dimensionality.* Hoboken(NJ): John Wiley and Sons.

Puterman, M., 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* NewYork: John Wiley and Sons.

Sanmartin, C., Gendron, F., Berthelot, J.-M. & Murphy, K., 2004. *Access to Health Care Services in Canada,2003,* Ottawa: authority of the Minister responsible for Statistics Canada.

Santibanez, P., Begen, M. & Atkins, D., 2007. Surgical block scheduling in a system of hospitals: an application to resource and wait list management in a British Columbia health authority. *Health Care Management Science,* 10(3), pp. 269-282.

Simester, D. I., Sun, P. & Tsitsiklis , J. N., 2006. Dynamic Catalog Mailing Policies. *Management Science,* 52(5), pp. 683-696.

Strum, D. P., May, J. H. & Vargas, L. G., 2000. Modeling the Uncertainty of Surgical Procedure Times: Comparison of Log-Normal and Normal Models.. *Anesthesiology,* 4(92), pp. 1160-1167.

Sutton, R., 1988. Learning to predict by the methods of temporal differences. *Machine Learning,* Volume 3, pp. 9-44.

Sutton, R. S. & Barto, A. G., 1998. *Reinforcement Learning.* Cambridge(MA): MIT Press.

Testi, A., Tanfani, E. & Torre, G., 2007. A three-phase approach for operating theatre schedules. *Health Care Management Science,* 10(2), pp. 163-172.

Tsitsiklis, J. N. & Van Roy, B., 2001. Regression methods for pricing complex American-style options. *IEEE Transactions on Neural Networks,* 12(4), pp. 694-703.

Tsitsiklis , . J. N. & Van Roy, B., 1997. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control,* 42(5), pp. 674-690.

`

Tsitsiklis, J. & Van Roy, B., 1996. Feature-based methods for large scale dynamic programming. *Machine Learning,* Volume 22, pp. 59-94.

Van Roy, B., Bertsekas, D. P., Lee, Y. & Tsitsiklis, J. N., 1997. A Neuro-Dynamic Programming Approach to Retailer Inventory Management. *roceedings of the 36th IEEE Conference on Decision and Control,* Volume 4, pp. 4052 - 4057.

Wald, A., 1947. *Sequential Analysis.* s.l.:John Wiley.

Watkins, C., 1989. *Learning from Delayed Rewards,* Cambridge,London: s.n.

White, D., 1963. Dynamic programming of Markov chains and the method of successive approximations. *Journal of mathematical analysis and application,* Volume 6, pp. 373-376.

Zhang, B., Murali, P., Dessouky, M. M. & Belson, D., 2009. A Mixed Integer Programming Approach for Allocating Operating Room Capacity. *Journal of the Operational Research Society,* Volume 60, pp. 663-673.