ABSTRACT

Title of Document: PRIORITIZING PATIENTS: STOCHASTIC DYNAMIC PROGRAMMING FOR SURGERY SCHEDULING AND MASS CASUALTY INCIDENT TRIAGE

William L. Herring
Doctor of Philosophy, 2011

Directed By: Associate Professor Jeffrey W. Herrmann
Applied Mathematics & Statistics and Scientific Computation Program
Department of Mechanical Engineering
Institute for Systems Research

The research presented in this dissertation contributes to the growing literature on applications of operations research models to problems in healthcare through the development and analysis of mathematical models for two fundamental problems facing nearly all hospitals: the single-day surgery scheduling problem and planning for triage in the event of a mass casualty incident. Both of these problems can be understood as sequential decision-making processes aimed at prioritizing between different classes of patients under significant uncertainty and are modeled using stochastic dynamic programming.

Our study of the single-day surgery scheduling problem represents the first model to capture the sequential nature of the operating room (OR) manager's decisions during the transition between the generality of cyclical block schedules (which

allocate OR time to surgical specialties) and the specificity of schedules for a particular day (which assign individual patients to specific ORs). A case study of the scheduling system at the University of Maryland Medical Center highlights the importance of the decision to release unused blocks of OR time and use them to schedule cases from the surgical request queue (RQ). Our results indicate that high quality block release and RQ decisions can be made using threshold-based policies that preserve a specific amount of OR time for late-arriving demand from the specialties on the block schedule.

The development of mass casualty incident (MCI) response plans has become a priority for hospitals, and especially emergency departments and trauma centers, in recent years. Central to all MCI response plans is the triage process, which sorts casualties into different categories in order to facilitate the identification and prioritization of those who should receive immediate treatment. Our research relates MCI triage to the problem of scheduling impatient jobs in a clearing system and extends earlier research by incorporating the important trauma principle that patients' long-term (post-treatment) survival probabilities deteriorate the longer they wait for treatment. Our results indicate that the consideration of deteriorating survival probabilities during MCI triage decisions, in addition to previously studied patient characteristics and overall patient volume, increases the total number of expected survivors.

PRIORITIZING PATIENTS: STOCHASTIC DYNAMIC PROGRAMMING FOR SURGERY SCHEDULING AND MASS CASUALTY INCIDENT TRIAGE

By

William L. Herring

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2011

Advisory Committee:
Associate Professor Jeffrey W. Herrmann, Chair
Associate Professor Elise Miller-Hooks, Dean's Representative
Professor Bruce Golden
Associate Professor Paul Smith
Associate Professor Paul Nagy

Dedication

To Ryland, without whose love, confidence, and sacrifice none of this would have been possible.

To Dosia, who brings unimaginable joy to each and every day.

Acknowledgements

First and foremost, I would like to extend my deepest gratitude to my advisor, Dr. Jeffrey Herrmann. Over the past three years he has devoted countless hours to guiding me, and at times pushing me, through the research presented in this dissertation. It is not an exaggeration to say that without his advice, insights, and perspective, I would still be wandering aimlessly through the graduate school wilderness.

For the past two and a half years, my research has been funded in part by an O.R. of the Future Grant (award number W81XWH-06-2-0057) from the U.S. Army Medical Research Acquisition Activity (administered by the University of Maryland Medical Center) and by a grant (award number 1H75TP000309-01) from the Centers for Disease Control and Prevention and the National Association of County & City Health Officials (administered by the Advanced Practice Center at the Montgomery County, Maryland Department of Health and Human Resources).

I have benefitted tremendously from collaborations with a whole host of people at the University of Maryland Medical Center (UMMC) in Baltimore. Dr. Paul Nagy and his Quality and Informatics Research team were instrumental in initiating the relationship and connecting us with the appropriate personnel throughout UMMC. I am particularly indebted to Sheila Gilger, Dr. James McGowan, Artanya Mills, Terell Thompson-George, Colleen Reilly, and everyone else associated with the operating room scheduling system, the Peri-Operative Services Department, and the Clinical Data Repository. Our research on mass casualty incident response planning also benefitted from the support and guidance of Michael Harrington, Director of the Patient Placement Center, Leonard Taylor, Jr., Vice President for Facilities Management, and Dr. James Chang, Emergency Response Manager.

I would also like to thank several members of the faculty and staff of the AMSC program. Dr. Bruce Golden was instrumental in my decision to pursue an undertaking of this magnitude, and he has continued to serve as a trusted mentor. Professors Raghu Raghavan, Michael Ball, Michael Fu, and Steven Gabriel exposed me to the breadth of topics that can be studied using operations research and equipped me with the mathematical tools I would need to do meaningful research. Both Alverda McCoy

and Dr. Konstantina Trivisa in the AMSC office helped me navigate the labyrinth of forms, proposals, and committee approvals required in order to actually complete my degree. Finally, I would like to thank Dr. Paul Smith and Dr. Elise Miller-Hooks for their willingness to take time out of their busy schedules to serve on my committee.

Table of Contents

List of Tables

List of Figures

List of Appendix Tables

Chapter 1. Introduction

The costs associated with the healthcare system have risen dramatically in recent years, and the increased public scrutiny to which the system has been subjected has been accompanied by increased attention from operations researchers and systems engineers (Valdez et al. 2010). Research in this area has touched on nearly all aspects of the healthcare system, with particular emphasis being given to problems in hospital operations management and public health administration. This dissertation contributes to this growing body of research through the development and analysis of mathematical models for two fundamental problems facing nearly all hospitals: the single-day surgery scheduling problem and planning for triage in the event of a mass casualty incident (MCI). While surgery scheduling is purely a hospital operations management problem, mass casualty response planning lies at the intersection of hospital management and public health disaster management. As the following chapters will demonstrate, both problems can be understood as sequential decision-making processes aimed on prioritizing between different classes of patients under significant uncertainty and can be modeled using stochastic dynamic programming (SDP). In addition to generating meaningful insights into the surgery scheduling and MCI triage problems, our analyses highlight the contributions that SDP models can make to the many sequential decision-making processes that permeate the delivery and management of healthcare.

1.1. Overview of Problems

The Single-Day Surgery Scheduling Problem

The problem of scheduling surgical procedures in hospital operating room (OR) suites has received extensive treatment in the operations research literature. The research presented in Chapters 2 through 4 contributes to these efforts by modeling and analyzing a fundamental, but previously understudied, interaction within surgery scheduling systems. In particular, our research is the first to explicitly model the transition from the generality of cyclical block schedules (which assign operating rooms to surgical specialties) to the specificity of the surgical schedule on the day of

surgery (which has individual patients assigned to specific ORs at specific times of day). Our analysis of this transition begins with a case study of the scheduling system at the University of Maryland Medical Center (Chapter 2) and continues with the development and analysis of a SDP model for the OR manager's decision-making process throughout this transition (Chapters 3 and 4). In particular, the case study reveals the importance of the OR manager's decisions to release unused blocks of OR time originally allocated to specific specialties and use them to schedule cases off of the surgical request queue. Our mathematical and computational results show how these decisions can be optimized using threshold-based block release and request queue policies.

Mass Casualty Incident Triage

The development of mass casualty incident (MCI) response plans and protocols has become a priority for hospitals, and especially emergency departments and trauma centers, in recent years. Research in this area is focused on how to effectively deliver life-saving medical care to a potentially large number of severely wounded casualties that will die if not treated promptly. Central to all MCI response plans is the triage process, which sorts patients into different categories in order to facilitate the identification and prioritization of those who should receive immediate treatment. While the majority of the medical literature on MCI triage is based on the personal experiences of trauma physicians and retrospective statistical analysis of past events, this area is beginning to receive more attention from operations researchers (see Chapter 5). Our research contributes to this trend by extending an existing model that approaches triage as a multi-class scheduling problem for "impatient" jobs (that is, jobs that will abandon the system prematurely if forced to wait too long for service). Our research in Chapter 6 incorporates into the existing models the important trauma principle that patients' long-term (post-treatment) survival chances deteriorate the longer they are forced to wait for treatment. Our results indicate that the consideration of deteriorating survival probabilities during MCI triage decisions increases the total number of expected survivors.

1.2. Stochastic Dynamic Programming in Healthcare

Healthcare decision-makers, especially in areas of hospital management and public health administration similar to those studied here, are rarely fortunate enough to have all necessary information made available to them at once. As a result, their decisions occur sequentially as information becomes available and situations around them change. As our analyses will demonstrate, SDPs are well-positioned to model these types of problems because of the explicitly sequential nature of the decision policies they produce. However, working with SDPs, and turning them into meaningful policy tools, can be potentially cumbersome for a number of reasons. First, the "curse of dimensionality" often means that realistic-sized problems are computationally intractable. Second, optimal SDP policies are often large, complex structures (reflecting the underlying state and decision spaces), and it can be difficult to turn these structures into meaningful insights that can be effectively communicated to decision-makers. This final point is particularly crucial in the area of healthcare, where intuitive and flexible solutions are needed in order to gain the support of stakeholders.

The research presented in this dissertation develops and tests exact and approximate solution procedures to SDPs that model the processes of sequential decision-making under uncertainty common to both of the problems described above. As will be discussed below, demand for elective surgery is naturally categorized according to the requesting surgical specialty (e.g., orthopedic) and the resulting management decisions focus on prioritizing between these categories in the face of the uncertain timing and quantity of this demand. In contrast, mass casualty incidents enforce no natural categorization, and triage during MCIs becomes a problem of sorting large numbers of patients into treatment categories and prioritizing between these categories. Our SDP formulations of these problems are no exception to the concerns raised above, and a recurring theme throughout our study of both problems is the use of special cases, theoretical structural results, and policy maps to generate meaningful insights that lead to high quality and intuitive heuristic procedures.

Chapter 2. Surgery Scheduling: Literature Review and Case Study

Research has shown that operating room (OR) scheduling plays a central role in determining hospital occupancy levels (McManus et al. 2003). Furthermore, the OR suite is known to be the most resource-intensive and profitable unit of a hospital (Macario et al. 1995). For these reasons, among others, the problem of scheduling surgical patients into operating rooms has received a great deal of recent attention in the operations research and management science literature. In a recent review of the literature on OR planning and scheduling, Cardoen et al. (2010) find nearly 250 manuscripts covering a large number of different problem variations, with over half of these contributions occurring in the last ten years. At its core, the surgery scheduling problem, in all its variations, involves the allocation of a fixed amount of resources (ORs, hospital staff) under uncertain demand. Like other scheduling problems, surgery scheduling approaches hope to make more efficient use of existing resources. However, as the literature and case study discussed below demonstrate, the large number of stakeholders and contentious nature of surgery scheduling introduce complexities that help to distinguish it from other scheduling problems.

The version of the surgery scheduling problem presented in this dissertation is focused on how operating room managers prioritize between different classes of surgical demand in the development of the schedule for a single day in an OR suite. In particular, this prioritization informs how managers schedule cases off of surgical waiting lists, or request queues. A case study of the surgery scheduling system at the University of Maryland Medical Center in Baltimore shows this prioritization to be part of a dynamic, sequential decision-making process, while the relevant literature either ignores this aspect of the problem or models it statically. The bulk of this chapter is devoted to a review of this literature and the results of the case study. These efforts indicate that surgery scheduling decisions must take into account surgeons' preferences and satisfaction in addition to traditional objectives such as maximizing OR utilization and reducing instances of overcapacity. The final section discusses

these insights and uses them to motivate our proposed model for the single-day surgery scheduling problem, which we present and analyze in Chapters 3 and 4.

2.1. Literature Review

Many hospitals schedule their OR suites using cyclic master, or block, surgery schedules, in which available OR space is assigned to specific surgical specialties, or service lines. For hospitals using block schedules, the literature on elective surgery scheduling describes the problem as consisting of three stages: (1) determining the amount of OR time to allocate to various surgical specialties, (2) creating a block schedule implementing the desired allocations, and (3) scheduling individual patients into available time (Blake and Donald 2002, Santibañez et al. 2007, Testi et al. 2007).

The first stage is referred to as case mix planning, and decisions at this stage typically reflect the long-term strategic goals of hospital management, such as meeting the demand for surgical specialties' services, achieving desired levels of patient throughput, or maximizing revenue (Blake and Carter 2002, Gupta 2007, Santibañez et al. 2007, Testi et al. 2007). The second and third stages represent medium- and short-term operational decisions, but differ markedly in their objectives. Block scheduling models have traditionally focused on implementing desired allocation levels (Blake and Donald 2002), but are moving toward a focus on leveling hospital bed occupancy and minimizing overcapacity (Beliën and Demeulemeester 2007, van Oostrum et al. 2008). Research on individual patient scheduling, including patient selection, room placement, and sequencing, typically aims to minimize patient delays or maximize OR utilization (Denton et al. 2007, Guinet and Chaabane 2003). The large number of stakeholders involved in surgery scheduling has also motivated a number of multi-objective models for both block scheduling and individual patient scheduling (Beliën et al. 2009, Blake and Carter 2002, Cardoen et al. 2009, Ozkarahan 2000).

Research on Block Schedules

As mentioned above, earlier research into creating block schedules focuses on implementing the allocation levels specified during case mix planning. Both Strum et

al. (1999) and Dexter et al. (1999b) use statistical analyses of historical hospital data to predict the number of hours surgical specialties should be allocated. In two related papers, Blake and Carter (2002) and Blake and Donald (2002) allocate available operating room time to different surgical specialties to maximize hospital profitability and meet surgeons' demand for OR space. Specifically, Blake and Carter use a goal programming model to determine the desired case mix, while Blake and Donald use integer programming to determine a block schedule that meets these specifications. Samanlioglu et al. (2010) use a similar integer programming approach to determine block schedules that meet surgeons' demand levels.

More recent research on block scheduling builds on the findings of McManus et al. (2003), which state that much of the variability in hospital bed occupancy levels is caused by imbalances in the surgical schedule. This research suggests that hospitals faced with overcrowding and high rates of patient diversion (i.e. patients being turned away due to lack of available beds) can reduce the occurrence of these problems by optimizing their block schedules. A subsequent wave of research addresses this issue by incorporating patients' lengths of stay into mathematical programming models and heuristic procedures aimed at leveling hospital bed occupancy and minimizing overcapacity (Beliën and Demeulemeester 2007, Chow et al. 2008, Price et al. 2011, Santibañez et al. 2007, Testi et al. 2007, van Oostrum et al. 2008). A final group of papers focuses on finding block schedules that minimize the amount of time patients have to wait for surgery (Tanfani and Testi 2010, Zhang et al. 2009).

Research on Individual Patient Scheduling

All hospitals, regardless of whether or not they use block schedules, must solve the problem of scheduling individual patients into specific OR time. For this reason, this stage of surgery scheduling benefits from a more robust literature than the earlier stages. Our review of this portion of the literature touches on the primary problem variations and methodologies. For a more thorough review, please see Cardeon et al. (2010). In general, patient scheduling studies can be categorized based on their consideration of the following three decisions: choosing which surgical cases to schedule, assigning cases to ORs on specific days, and sequencing cases within

specific ORs. Typically either the first two or last two of these decisions are modeled, although some research focuses more narrowly on just one of these decisions. Based on which subset of the decisions is being considered, scheduling objectives range from minimizing patients' waiting times to maximizing OR utilization and reducing overtime. While each of the studies considers different sets of realistic constraints (such as the consideration of an underlying block schedule, recovery and downstream bed availability, limitations on patient waiting times, and surgeons' preferences) and different levels of stochasticity (with respect to case durations), the primary objectives remain fairly consistent throughout.

Ozkarahan et al. (2000) use an integer goal programming approach to select which patients to schedule and in which ORs to schedule them on a single day. Testi et al (2007) study a problem with a similar scope using discrete event simulation to judge the quality of different scheduling policies. Other research that focuses on selecting which patients and which ORs does so over the course a longer planning horizon (typically one week) (Ogulata and Erol 2003, Lamiri et al. 2008a,b, Min and Yih 2010). Each of these papers relies on a multi-stage model to address the separate decisions of choosing which patients and then assigning them to specific ORs on specific days. Ogulata and Erol (2003) use a hierarchical mathematical programming approach, while Lamiri et al. (2008a) use a column generation algorithm. Both Lamiri et al. (2008b) and Min and Yih (2010) use two-stage stochastic programming techniques.

Most of the research that focuses on assigning patients to ORs and sequencing the cases within ORs focuses on the single day problem. The two-stage nature of the problem again necessitates sophisticated heuristics or multi-stage solution procedures. Sier et al. (1997) formulate the problem as a nonlinear integer program and use simulated annealing to obtain good solutions. Jebali et al. (2006) solve a series of integer programs. Pham and Klinkert (2008) model the problem as a job shop scheduling problem and find solutions using mixed integer linear programming. Hans et al. (2008) use off-line bin-packing heuristics to create robust schedules using planned slack, and their work is the exception in that it schedules cases over the course of a week rather than a single day.

Other research is more targeted in its focus. Guinet and Chaabane (2003) model the assignment of patients to operating rooms (without regard to sequencing) as a generalized assignment problem and use a primal-dual algorithm to find high quality solutions. Denton et al. (2010) assign cases to operating rooms using a stochastic programming model to incorporate uncertain case durations. Both Denton et al. (2007) and Cardoen et al. (2009) investigate the optimal sequencing of cases within an OR using stochastic linear programming and a branch-and-price approach, respectively.

An Understudied Interaction

A fundamental, but understudied, element of the day-to-day job of surgery scheduling is the transition from the generality of the block schedule (in which OR time is allocated to specialties) to the specificity of a completed schedule for a particular day (in which OR time is assigned to specific cases). While block schedules are often incorporated as constraints in the individual patient scheduling models discussed above (Hans et al. 2008, Min and Yih 2010, Pham and Klinkert 2008, Testi et al. 2007), each of these models schedules large batches of patients all at once, rather than sequentially. As will be discussed in greater detail below, in practice, individual patients are scheduled into ORs over time as the demand for surgery is generated, resulting in a dynamic and sequential decision-making process. The few studies that do consider the dynamic evolution of a surgical schedule focus on the scheduling of add-on, or waiting list, cases, but do so either for a limited number of cases or on a limited number of days (Dexter et al. 1999a, Dexter and Traub 2002, Dexter et al. 2003, Dexter and Macario 2004, Gerchak et al. 1996). As a result, these studies provide only a limited picture of the dynamics of the surgery scheduling process. The fundamental contribution of our treatment of the single-day surgery scheduling problem in the following chapters is the explicit modeling of the dynamic transition between the block and individual patient stages of surgery schedule and the management policies that control this interaction.

Related Research

The stochastic dynamic programming (SDP) model we propose in Chapter 3 has much in common with existing research on capacity allocation (Schütz and Kolisch 2010a,b) and, in particular, airline revenue management (Brumelle and Walczak 2003, Lee and Hersh 1993, Subramanian et al. 1999). While the specifics of our model will be presented later, we conclude our literature review with a brief review what distinguishes our work from this related research. In these problems, a finite resource (OR time, seats on a flight) with a fixed expiration date (the day of surgery, the departure time for the flight) must be allocated to competing demand classes. The demand from each of the classes arrives over time, and decision-makers must decide how much of the resource to allocate to lower priority classes and how much to reserve for higher priority classes. In the existing models, arriving demand must be accepted or rejected at the moment of its arrival and rejected demand is lost. In the surgery scheduling problem, however, lower priority demand is placed on a waiting list, or request queue, and can be accepted at a number of different decision points leading up to the day of surgery. While our proposed surgery scheduling model displays solution behavior similar to the revenue management models (particularly to Lee and Hersh 1993), the introduction of the request queue concept increases the complexity of the state space and complicates the analysis leading to these results.

2.2. Case Study of a Surgery Scheduling System

In order to gain insight into how the transition from the block schedule to individual patient scheduling occurs in a real hospital's operating room suite, we continue with the results of a case study of the surgery scheduling system at the University of Maryland Medical Center in Baltimore. The case study consists of two components: (1) information about the system gathered through meetings and interviews with administrators and schedulers in the hospital's Peri-Operative Services department and (2) data collected during a detailed observation of the evolution of the schedule for a particular day in the OR suite. The first stage serves to identify the fundamental components and decision-makers in the scheduling system, from which we are able to build a model of how surgical cases flow through this

system. The second stage illustrates how these processes and decisions occur in practice and, in particular, leads to a greater understanding of the factors influencing an OR manager's decision-making throughout the development of the schedule.

A Model for the Scheduling System

The University of Maryland Medical Center (UMMC) uses a cyclic block schedule to allocate operating room time to particular surgical specialties and surgeons for their elective surgeries. The block schedule is the primary mechanism by which UMMC guarantees access to OR time for its surgeons, who can be viewed as customers of the hospital in that they are free to take their surgical cases to another hospital if they are unsatisfied with the OR time they are allocated. Table 1 shows what this block schedule looks like for a subset of the ORs at UMMC during the spring of 2009. The bulk of the block schedule is cyclic on a weekly basis, with many of the ORs (e.g., 8, 17, and 20) being allocated to the same specialty every day of the week and others changing specialties from day to day (e.g., 18 and 29). As shown in rooms 21 and 22 on Wednesday, some blocks are split into morning and afternoon sessions and allocated separately. Room 22 also provides an example of how a room can be allocated to different specialties in alternating weeks (such as the 1st, 3rd, and 5th Monday of each month versus the 2nd and 4th Monday of each month). Blocks marked as "URGENT" or "OPEN" are unallocated and made available for emergency surgeries and for specialties and surgeons that do not have an allocated block on the day in question, respectively.

The first step in understanding the impact that the block schedule has on the development of specific daily schedules is to identify the critical components of the scheduling system and the ways in which surgical cases flow through this system. It is important to distinguish here between the flow of surgical case information through the scheduling system (an *information* system) and the flow of patients through the different units of a hospital (a *physical* system). Our focus will be on the flow of surgical cases through the scheduling system.

Table 1. The cyclic block schedule for a subset of operating rooms at UMMC in spring 2009

Room	Monday	Tuesday	Wednesday	Thursday	Friday
8	Urology	Urology	Urology	Urology	Urology
17	Neurosurgery	Neurosurgery	Neurosurgery	Neurosurgery	Neurosurgery
18	Surgical Oncology	Thoracic	General Surgery	Transplant	General Surgery
20	Pediatrics	Pediatrics	Pediatrics	Pediatrics	Pediatrics
21	Cardiac	Vascular	Donor (am) OPEN (pm)	Vascular	Donor
22	URGENT (1,3,5) Transplant (2,4)	Cardiac	Transplant (am) OPEN (pm)	Cardiac	Transplant (1,3,5) URGENT (2,4)
29	General Surgery	General Surgery	Surgery Oncology	Surgical Oncology	Surgery Oncology

Demand for surgery is generated when it is determined (by a physician in a clinic, a surgeon making his rounds, or emergency personnel, for instance) that a patient requires surgery. There are two primary ways that the hospital and the OR staff become aware of this demand. If the surgery is urgent, the patient is taken to an OR as soon as possible, and the information about the case essentially bypasses the scheduling system (although it is entered later for documentation and billing purposes). On the other hand, if the surgery is deemed to be elective, the details of the case are communicated to the hospital by the surgical specialty (or the surgeon) associated with the case. Therefore, the surgical specialties serve as the entry point for surgical cases into the elective surgery scheduling system, as illustrated in Figure 1 below.

At UMMC, demand for elective surgery is day-specific. That is, when the demand is communicated to the scheduling system, it is accompanied by a specific date in the future on which the surgeon would like to perform the surgery. This day-specific demand goes hand in hand with the cyclic block scheduling approach, which guides the surgeon's choice of dates and provides a measure of assurance (at least for

specialties with allocated blocks) that cases can be scheduled on the desired date. This notion of day-specific demand and its relationship to the block schedule allows us to isolate the scheduling system for a particular day in the OR suite. Another fundamentally important aspect of the demand for elective surgery on a particular day is that it is generated over time rather than all at once. According to UMMC staff, surgical specialties begin generating demand approximately one month before the day of surgery and continue to generate elective demand up until the day before surgery. Demand generated on the day of surgery, which is essentially demand for immediate surgery, is classified as urgent rather than elective.

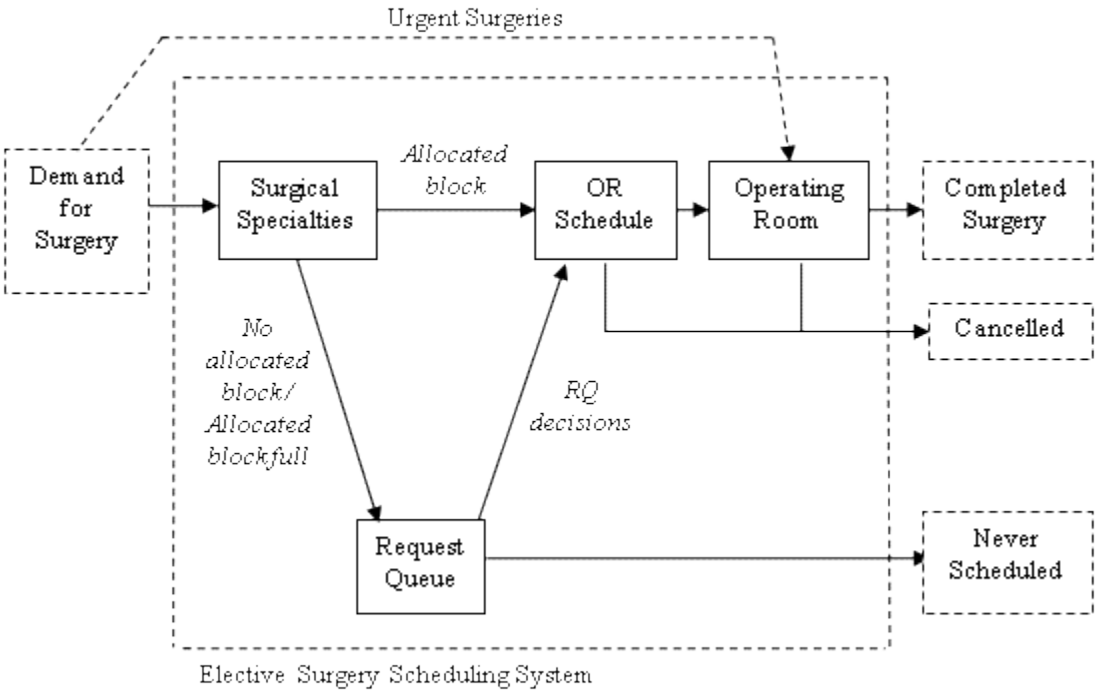

Figure 1. Flow of surgical cases through UMMC's scheduling system

If we focus our attention then on the scheduling system for a single day, we can model the flow of surgical cases through the system using the diagram in Figure 1. The available OR time on the day in question has been allocated to specific surgical specialties according to the block schedule. We refer to specialties on this day's block schedule as "primary" specialties and their demand for surgery as "primary" demand. Initially, primary specialties have complete control of their allocated blocks and, as indicated in Figure 1, primary demand associated with an allocated block is added to

the OR schedule without any further input from hospital administrators or scheduling staff. Specialties (or surgeons) that do not have allocated blocks ("secondary" specialties) can still generate demand for the day in question ("secondary" demand), but they must submit their cases to a surgical request queue (RQ). If a primary specialty's allocated OR time has been filled, it may also submit excess demand to the RQ. Just as demand is day-specific, the surgical RQ at UMMC is day-specific and unmet requests are not automatically rolled over from one day of surgery to the next.

In the period leading up to the day of surgery, cases accumulate on the RQ and the OR manager looks to schedule these RQ cases into OR time that has not been filled by the primary specialties. As illustrated in Figure 1, adding a RQ case to the OR schedule requires an active RQ decision on the part of the OR manager. This sits in direct contrast to the scheduling of primary demand into allocated blocks, which is controlled by the primary specialties and is outside the OR manager's control. Recalling that demand for surgery is generated over time, it is clear that the OR schedule for the day of surgery evolves over time as the day of surgery approaches. The OR manager's RQ decisions interact directly with the timing and volume of the primary demand to determine which cases will be added to the schedule.

At UMMC, there are fixed days before surgery, referred to as *block release dates*, after which OR managers can begin scheduling RQ cases into unfilled blocks. The block release dates (typically two or three days before the day of surgery) vary from specialty to specialty, with one of the chief reasons cited for this variation being the differences in primary demand patterns between the specialties. Before the block release dates, RQ cases may not be assigned to open times. On each day between the block release date and the day of surgery, the OR manager uses the *request queue policy* to determine how and when to schedule RQ cases into open times. The factors that contribute to and influence the RQ policy will be discussed in more detail in the next subsection.

Finally, Figure 1 also illustrates how the cases that enter the elective surgery system eventually exit the system. On the day of surgery, cases on the OR schedule move to the operating rooms and exit the system as completed cases. Cases on the schedule can also exit the system by being cancelled (or rescheduled for another day).

Cancellations that occur before surgery begins exit the system directly from the OR schedule, while some cancellations happen after the case has entered the OR. Lastly, some cases on the RQ never get added to that day's OR schedule and exit the system directly from the RQ. Because the RQ at UMMC is day-specific, the surgeons associated with unmet demand must resubmit their cases to the surgery scheduling system for another day in the future, where they appear as newly generated demand.

The Development of a Single Day's Schedule

From this model of the surgery scheduling system, we can see that the final schedule for a given day in the OR suite at UMMC is a product of the interaction of the underlying block schedule, the primary demand patterns for the specialties with allocated blocks, the block release dates associated with each of these specialties, and the RQ policies used by the OR manager to schedule RQ cases into unused time. As discussed in the literature review above, none of the mathematical models in the existing research fully capture all the aspects of this interaction. In order to build such a model, we must take a closer look at the factors contributing to the OR manager's decision-making policies. To this end, we proceed with a discussion of how the schedule for a specific day in the spring of 2009 was developed.

The operating room suites at UMMC consist of 19 rooms in the General OR Suite, four rooms in the North OR Suite, six rooms in the Shock Trauma Center, and two additional minor ORs. The trauma rooms are reserved for urgent and emergency surgeries, and thus are not included in the block schedule. While the minor ORs appear on the block schedule, they are too small for most elective surgeries and are therefore subject to strict scheduling restrictions. Therefore, we focused our observation of UMMC's scheduling system on the 23 rooms in the General and North OR Suites, each with a stated capacity of eight hours per day. On the day in question, these rooms were allocated to surgical specialties according to the block schedule shown in Table 3 below.

Working closely with the Peri-Operative Services department and the OR manager at UMMC, we tracked each surgical case that entered the scheduling system for the day in question. For each case, we collected basic information such as the

associated specialty and surgeon, standard procedural codes, the expected duration, and special resource requirements. In addition, we tracked the movement of each case through the system from the time it arrived to the time it exited. This data included the scheduling lead (how many days before the day of surgery it arrived), transitions from the RQ to the schedule, swapping of scheduled cases between ORs, case cancellations, and the ultimate completion of the case on the day of surgery. In addition, to the extent possible, we recorded the reasoning behind scheduling decisions made by the OR manager, with a particular focus on why and when RQ placement decisions were made (or not made).

The earliest arriving case for the day in question was added to the schedule 30 working days before the day of surgery, and demand continued to arrive up until the night before surgery. Using Figure 1 to identify the different trajectories through the scheduling system, Table 1 shows the frequencies of each possible trajectory. Overall, 69 surgical cases entered the system for the day in question, of which 51 ultimately received surgery, eleven were scheduled and later cancelled or rescheduled, and seven were never scheduled. Roughly two-thirds of the completed cases were generated by the primary specialties, while nearly all of the cancelled cases came from the primary specialties. Of the cancelled cases, seven were for clinical reasons (such as the need for further testing), two were rescheduled, one was due to a lack of recovery beds, and the other was cancelled by the patient after entering the OR.

Table 2. Frequency of trajectories through the scheduling system for a day in UMMC's OR suite

Trajectories Through the Scheduling System	Frequency
Total Cases	69
Completed Cases	
Specialty → Schedule → Operating Room → Completed	35
Specialty → Request Queue → Schedule → Operating Room → Completed	16
Cancelled or Rescheduled Cases	
Specialty → Schedule → Operating Room → Cancelled	0
Specialty → Request Queue → Schedule → Operating Room → Cancelled	1
Specialty → Schedule → Cancelled	9
Specialty → Request Queue → Schedule → Cancelled	1
Cases Never Scheduled	
Specialty → Request Queue → Never Scheduled	7

Table 3 presents summary statistics for the primary demand generated by each of the specialties on the block schedule on the day of our observation. Of the 20 rooms allocated to specialties, only four rooms had primary specialties that generated no demand for the day in question. For the other rooms, the total primary demand is shown both as the number of cases and the sum of the scheduled durations. Most of the primary specialties scheduled over six hours of cases into their allocated blocks, with several specialties exceeding the stated capacity with more than ten hours of cases. The average scheduling lead for the primary specialties ranged from one day before surgery (Cardiac) to 23 days (Otolaryngology), and seven specialties had average scheduling leads of one week or less.

Because the block release date for each room indicates the day on which the OR manager may begin scheduling RQ cases into unused time, Table 3 also shows the portion of the primary demand that arrived after the block release date. As expected, the same specialties that had short scheduling leads generated most of their demand after their block release date. As a closer look at the secondary demand reveals, these specialties with late-arriving demand ran the risk of having their allocated blocks given to RQ cases by the OR manager.

Secondary demand can divided into two categories. Primary specialties with allocated blocks often assign their blocks to particular surgeons. Therefore, some secondary demand is generated by specialties that have allocated blocks, but by surgeons that have not been assigned a block on the day in question. The rest of the secondary demand comes from specialties that do not have allocated blocks on the day in question. Table 4 presents a summary of the secondary demand for each of these categories on the day of our observation. As the table shows, the majority of the secondary demand came from specialties that already had allocated blocks. On average, these requests for surgery were added to the RQ around four days before the day of surgery. The OR manager first considered placing RQ cases into the OPEN rooms (rooms 7 and 14) one week before the day of surgery (two days prior to the first block release date), and we can see that roughly a fifth of the secondary demand was generated before the first RQ decision was made. More detailed data on the

timing of both the primary and secondary demand for the day of our observation is presented in Table 5.

Table 3. Block schedule allocations and summary of primary demand for elective surgery for a day in UMMC's OR Suite

OR	Primary Specialty	Block Release Date (days)	Total Primary Demand		Primary Demand Scheduled After Block Release		Average Scheduling Lead (days)
			# of Cases	Hours	# of Cases	Hours	
7	OPEN						
8	Urology (SUR)	3	3	6.3			19.0
9	Orthopedics (SOR)	0	0	0			
10	Orthopedics (SOR)	2	0	0			
11	Neurosurgery (SNG)	2	3	8.9			9.0
12	Oral / Dental (DOM)	3	1	10.0			14.0
14	OPEN						
15	Otolaryngology (SEN)	3	4	11.6			23.3
16	Orthopedics (SOR)	2	4	9.1			13.0
17	Neurosurgery (SNG)	2	3	12.5			3.8
18	Transplant (STO)	3	2	6.3			10.0
19	Oncology (SON)	3	5*	17.2*	2	6.3	3.8
20	Pediatrics (SPD)	2	4	5.4			5.0
21	Cardiac (SCS)	2	3	13.9	3	13.9	1.0
22	Vascular (SVA)	2	0	0			
23	Thoracic (STH)	2	3	6.5	2	5.3	2.3
24	Cardiac (SCS)	2	0	0			
25	Cardiac (SCS)	2	1	5.6	1	5.6	1.0
26	URGENT						
29	Oncology (SON)	3	3	6.9			10.7
30	Gynecology (OGY)	3	1	1.2	1	1.2	1.0
31	General Surgery (SGL)	3	2	6.1			9.0
32	General Surgery (SGL)	3	2	6.4			9.0

* 3 cases (10.8 hours) were cancelled or rescheduled before the final 2 cases were scheduled.

Table 4. Summary of secondary demand for elective surgery for a day in UMMC's OR suite

	Total Secondary Demand		Demand Before First RQ Decision		Average Request Lead (days)
	# of Cases	Hours	# of Cases	Hours	
Specialties With an Allocated Block	24	76.6	5	23.0	4.3
Specialties Without an Allocated Block	2	6.1	0	0	3.4

Table 5. Number of cases and hours of primary and secondary demand for a day in UMMC's OR suite

		Days Before Surgery																
Category of Demand		>15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Primary																		
OR	Specialty																	
8	SUR	1 (1.5)		2 (4.9)														
9	SOR																	
10	SOR																	
11	SNG						1 (3.5)	1 (1.5)				1 (3.9)						
12	DOM			1 (10)														
15	SEN	4 (11.6)																
16	SOR	2 (5.0)							1 (1.8)						1 (2.3)			
17	SNG									1 (2.7)	1 (5.2)			1 (4.6)				
18	STO							2 (6.3)										
19	SON								1 (5.1)					2 (5.8)			2 (6.3)	
20	SPD												4 (5.4)					
21	SCS																3 (13.9)	
22	SVA																	
23	STH												1 (1.2)				2 (5.3)	
24	SCS																	
25	SCS																1 (5.6)	
29	SON			1 (1.7)					2 (5.2)									
30	OGY																1 (1.2)	
31	SGL									2 (6.1)								
32	SGL									2 (6.4)								
Secondary																		
Specialties With an Allocated Block		1 (12.5)	1 (3.8)		2 (5.0)				1 (1.7)				1 (3.3)	2 (2.7)	1 (2.0)	3 (6.9)	11 (36.9)	3 (4.5)
Specialties Without an Allocated Block														1 (4.1)	1 (2.0)			

Note: Demand arrival is presented as "Number of cases (hours of cases)." The solid vertical bars for the primary demand represent the timing of the block release date for each specialty. The solid vertical bar for the secondary demand represents the timing of the first RQ decision.

Also shown in Table 5 are the block release dates for each of the primary specialties (the dark vertical bars for each OR) and the first day on which RQ cases were considered for placement in unused time (the dark vertical bar for the secondary demand). This detailed demand information allows us to track exactly which RQ decisions were feasible and which cases were actually scheduled on each day leading up to the day of surgery. In order for a RQ decision (placing a specific case in a specific OR) to be feasible, we require the OR to have already released its block and be able to accommodate the case without exceeding eight scheduled hours. In certain instances, the feasibility of a room changed from day to day based on case cancellations or swaps between rooms. We will point out these instances when appropriate, but will avoid going into greater detail in order to maintain focus on the RQ decisions.

As mentioned above, the OR manager first considered placing RQ cases into the OPEN rooms five days before the day surgery. Table 6 and Table 7 show the state of the RQ on the five days leading up to the day of surgery and on the day of surgery, respectively. For each case, the corresponding specialty and expected duration are shown, in addition to an indicator of whether this was the first day on which this case was considered for addition to the schedule. The tables also show the ORs that could feasibly take on at least one of the RQ cases (based on the block release dates and a stated capacity of eight hours per day) and the subsequent RQ decisions for each case on the day in question. RQ cases that are not scheduled on one day are reconsidered for placement the following day (see, for example, Case 6 on days 4 through 2), and we correspondingly refer to these RQ decisions as "deferrals."

On day 5, a concerted effort was made to schedule each of the cases that were already on the RQ, with two cases being scheduled in apparent contradiction with stated block release dates and RQ policies. One of the exceptions was due to the resolution of surgical equipment restrictions (Case 1 into Room 30), while the other was prompted by a match between the secondary specialty and the original room allocation (Case 5 from Orthopedics into Room 16). Relatively few RQ cases were added to the schedule over the next four days, in spite of the increased feasibility resulting from the block releases on days 3 and 2. Over the course of these four days,

room 7 was filled by two additional cases and another case was added to room 8, reflecting another match between the secondary specialty (Urology) and the original room allocation.

Table 6. State of the surgical request queue and subsequent RQ decision over the five days before the day of surgery

Days Before Surgery	Specialty	Duration (hours)	New?	Feasible Rooms (hours filled)	Decision
Day 5					
Case 1	OGY	3.8	y		Scheduled in Room 30*
Case 2	SEN	12.5	y		Scheduled in Room 14
Case 3	DOM	2.0	y	Rooms 7, 14 (0)	Scheduled in Room 7
Case 4	DOM	3.0	y		Scheduled on another day
Case 5	SOR	1.7	y		Scheduled in Room 16*
Day 4					
Case 6	SUR	3.4	y	Room 7 (3.8)	Deferred
Day 3					
Case 6	SUR	3.4	n		Deferred
Case 7	EYE	4.1	y	Room 7 (3.8), Room 8 (6.4), Room 18 (6.3), Room 31 (6.1), Room 32 (6.4)	Scheduled in Room 7
Case 8	SGL	1.4	y		Deferred
Case 9	SGL	1.4	y		Deferred
Day 2					
Case 6	SUR	3.4	n		Scheduled in Room 8
Case 8	SGL	1.4	n	Room 7 (3.8), Room 8 (6.4), Rooms 10, 21, 22, 24, 25 (0), Room 18 (6.3), Room 20 (5.4), Room 23 (1.2), Room 31 (6.1), Room 32 (6.4)	Deferred
Case 9	SGL	1.4	n		Deferred
Case 10	SEN	2.0	y		Deferred
Case 11	EYE	2.0	y		Scheduled in Room 7
Day 1					
Case 8	SGL	1.4	n		Deferred
Case 9	SGL	1.4	n		Deferred
Case 10	SEN	2.0	n	Rooms 10, 21, 22, 24, 25 (0), Room 18 (6.3), Room 20 (5.4), Room 23 (1.2), Room 31 (6.1), Room 32 (6.4)	Removed from RQ
Case 12	SGL	1.5	y		Deferred
Case 13	SGL	3.0	y		Deferred
Case 14	SGL	2.4	y		Deferred

* Case 1 was placed on the RQ due to surgical equipment restrictions in Room 30, which were later resolved. Case 5 was given an exception and scheduled in Room 16 in advance of the block release date.

Of particular interest on days 2 and 1 was the decision by the OR manager not to place any of the waiting RQ cases into the empty ORs (rooms 10, 21, 22, 24, and 25). The reason the manager gave for these deferrals was the anticipation of late-arriving

primary demand from the specialties controlling these rooms. In two of the five rooms (rooms 21 and 25, both Cardiac rooms), this anticipation was justified by the arrival of significant primary demand on the day before surgery. As shown in Table 7, secondary demand from the specialties controlling two of the other rooms (rooms 10 and 24, allocated to Orthopedic and Cardiac, respectively) arrived on the day before surgery. Again showing a preference for matching up secondary specialties with original room allocations, these RQ cases were scheduled in the respective ORs on the morning of surgery.

Table 7. Status of the surgical request queue and subsequent RQ decisions on the day of surgery

Day of Surgery	Specialty	Duration (hours)	New?	Feasible Rooms (hours filled)	Decision
Day 0					
Case 8	SGL	1.4	n	Rooms 9, 10, 22, 24 (0), Room 18 (6.3), Room 19 (6.3), Room 20 (5.4), Room 23 (6.5), Room 25 (5.6), Room 29 (6.9), Room 31 (6.1), Room 32 (6.4)	Never scheduled
Case 9	SGL	1.4	n		Never scheduled
Case 12	SGL	1.5	n		Scheduled in Room 10
Case 13	SGL	3.0	n		Scheduled in Room 19
Case 14	SGL	2.4	n		Scheduled in Room 29
Case 15	SCS	1.9	y		Scheduled in Room 24
Case 16	SCS	7.2	y		Scheduled in Room 24
Case 17	SOR	3.5	y		Scheduled in Room 10
Case 18	SPD	0.4	y		Scheduled in Room 20
Case 19	SPD	1.7	y		Scheduled in Room 20
Case 20	SON	7.3	y		Scheduled in Room 14**
Case 21	STO	4.4	y		Scheduled in Room 22
Case 22	STO	4.4	y		Never scheduled
Case 23	STH	3.4	y		Scheduled on another day
Case 24	SCS	1.9	y*		Scheduled in Room 25
Case 25	SCS	1.9	y*		Scheduled in Room 24
Case 26	SGL	0.8	y*		Scheduled on another day

* These cases arrived to the RQ overnight on the night before surgery, and were therefore still considered elective cases.

** Cases were swapped between rooms 9, 14, and 16, freeing up Room 14 and filling up Room 9.

After these specialty-based RQ decisions, three rooms were still empty on the morning of surgery (room 9, the final block to be released, room 22, and room 26, originally marked URGENT) and several other rooms had an available hour or two.

As many of the remaining RQ cases were scheduling into these open times as possible, with some continued preference for matching up secondary specialties with original block allocations (scheduling Pediatric cases into room 20 and swapping cases between rooms so that Orthopedic cases could be done in room 9). After all RQ decisions were made on the day of surgery, every OR with the exception of room 22 (which had 4.4 hours scheduled) and room 26 (which continued to be empty) had at least 6 hours of surgeries on its schedule. Room 26 remained empty until a case scheduled into a different OR was moved there late in the afternoon on the day of surgery.

From this data on the RQ decisions made by the OR manager, and their relationship to the primary and secondary demand arrival patterns, we begin to get a sense of the underlying RQ policies guiding the decisions. The decision to place a RQ case into an OR with available time is largely influenced by the anticipation (or lack thereof) of late-arriving demand from the primary specialty, showing a reluctance to give away allocated block time if the primary specialty is likely to end up needing it. Once it has been deemed appropriate to add RQ cases to a room's schedule, there is a clear preference for matching the secondary specialty to the primary specialty originally controlling the room. In the following section we will discuss these observations in more detail, while relating them to both the existing literature on block release policies and to our proposed model.

2.3. Discussion and Modeling Implications

In discussing the results of our case study, it is helpful to return to our initial motivations for studying the surgery scheduling system. As the literature review in Section 2.1 illustrates, very little work has been done on the transition from the block scheduling stage to the individual patient stage of surgery scheduling. Our initial meetings with administrators and schedulers at UMMC identified the block release dates as the fundamental tool that makes this transition work. Furthermore, UMMC had recently gone through significant changes in their block release policy. In the year prior to our case study, the block release dates had been moved from their current positions back to a week before the day of surgery. However, this change was

not well received, and the block release dates were subsequently returned to their original (and current) positions. These internal issues at UMMC demonstrated the contentious nature of block release decisions and, along with the gap in the literature, motivated the very natural question of how block release dates can be set optimally.

A series of papers by Dexter et al. (2003) and Dexter and Macario (2004) actually studies this very question, but in a limited way. In these papers, the authors study the addition of a single, elective add-on (i.e., request queue) case to an existing surgical schedule on block release dates ranging from one to five days before the day of surgery. In their analyses, the RQ case is added to the OR with the largest amount of unused time at the moment of the block release, and the efficiency of the resulting OR schedules is compared via simulation. Based on their results, they conclude that the timing of the block release has very little impact on the efficiency of the final schedule and suggest that hospitals set their block release policies according to the preferences of their particular stakeholders.

However, the general applicability of their work is limited by a number of factors. First, they assume that the block schedule underlying the development of the schedule has been allocated based on optimizing efficiency (according to the methodologies discussed in Strum et al. (1999)). Second, they only consider the addition of one RQ case to the schedule. They justify the limitation to just one RQ case by arguing that if the underlying block schedule has been allocated optimally, the number of RQ cases for a particular day of surgery will rarely exceed one case. Finally, in their simulations, they do not consider the reality that adding a RQ case to the schedule will dynamically influence the evolution of the schedule after the block release.

In practice, hospitals may have good reasons for not allocating their blocks using the methods advocated by Strum et al. (1999). For instance, at UMMC, which is a large tertiary care center, some blocks must be allocated to high priority, low volume specialties (such as Transplant) to ensure that cases from these specialties can be scheduled and performed promptly when they are generated. Furthermore, as the case study demonstrates, OR managers frequently have to consider more than one RQ case, and the impact that RQ placements have on the development of the rest of the schedule is central to their decision-making. In the discussion of their results, Dexter

and Macario (2004) acknowledge that the impact of the block release timing increases when the OR suite is likely to have several ORs with unused time or when the RQ cases are longer (both of these situations were observed during our case study at UMMC). It is logical, then, to question whether the conclusions of Dexter et al. (2003) and Dexter and Macario (2004) continue to hold in more general settings.

Focus on Block Release Timing

Returning to the results of our case study, we see that there are two fundamentally different types of decisions that the OR manager must make in order to add a RQ case to the schedule. The first involves the decision to release an allocated block, and the second involves the selection of a particular RQ case to schedule in the released time. The research on block release timing discussed above pertains to the first of these decisions, and our research will correspondingly continue to focus on this aspect of the problem.

In the case study, the selection of individual RQ cases was based on a range of factors, including time spent on the RQ (reflecting a first-come, first-served prioritization), specialty and equipment matching, and the relative urgency of the case, to name a few. While complex, these factors tended to be easily identifiable and were more reflective of the particulars of each case than the structure of the scheduling system as a whole. In contrast, the decision to release an allocated block in the first place was more dependent on system-wide factors, such as the primary demand arrival patterns and the balance between waiting RQ cases (as a group rather than individually) and yet-to-arrive primary cases. As such, our modeling efforts will focus on the factors and interactions within the scheduling system (as depicted in Figure 1 and illustrated by the case study) that contribute to block release decisions. Our decision not to model the particular factors associated with each individual RQ case (which would be impossible to model in full) facilitates our approach in two ways. First, it makes our model more parsimonious and helps maintain our focus on the decisions to release allocated blocks (in essence choosing how many RQ cases to add the schedule, rather than which ones). Second, from an implementation

standpoint, it allows the OR manager to retain a degree of flexibility in responding to the unique details and requirements of particular cases.

Before continuing with the framework with our modeling approach, it is important to draw a distinction between block release dates and the decision to release an allocated block. As illustrated in several of the ORs during our observation at UMMC, an allocated OR with unused time is not automatically released just because the block release date has passed (see Table 6 above). Furthermore, not all of the unused time in a block must be released at once. As acknowledged by one of the administrators at UMMC, block release dates are merely constraints on the OR manager's ability to make block release (and subsequent RQ) decisions. As a result of these observations, in the following chapters, our modeling objective focuses not on choosing optimal block release dates but rather on optimizing the timing and extent of block release and RQ decisions over the days leading up to the day of surgery. This broader scope allows us to more realistically model the block release and request queue decisions made by an OR manager over the development of a particular schedule.

Modeling Framework

The reasoning behind the block release and request queue decisions made by the OR manager during our observation at UMMC make it clear that the primary demand arrival patterns play an important role in these decisions. In choosing not to schedule RQ cases into rooms 10, 21, 22, 24, and 25 on the two days prior to the day of surgery (even though their block release date had passed), the manager made it clear that it was more desirable to make the current RQ cases wait for a decision than to block any late-arriving primary cases from accessing their allocated blocks. In two of these five rooms, this anticipation was rewarded by the last minute arrival of primary demand, while the other three rooms were ultimately released to RQ cases on the day of surgery. The decisions with respect to these rooms were not based on maximizing the utilization of the ORs (the manager chose uncertain future demand over existing, known demand), but instead were based on customer satisfaction costs associated with the competing demand types.

In their research, Dexter and Macario (2004) acknowledge these satisfaction costs, but do not incorporate them directly into their decision framework. In order to mathematically model block release decisions in a more general setting, it is necessary to formalize these costs and analyze how they interact with the primary arrival patterns to influence the OR manager's decisions. Suppose that an OR has time for one additional case and the RQ contains a case that fits in this available time. The OR manager is faced with the following choice: either release the block and schedule the RQ case in the available time or defer scheduling the case (and considering it again on the following day). Deferring the RQ case leaves open the possibility that a primary case may arrive to use the remaining time, but is undesirable because the surgeon and the patient associated with the RQ case must wait at least one more day for a decision. We define this satisfaction cost as a *deferral cost*. If the RQ case is scheduled (thus filling the OR) and another primary case arrives, then the OR manager has blocked the primary specialty's access to its allocated room. We define this satisfaction cost as a *blocking cost*. The balance between deferral costs and potential blocking costs informs the OR manager's decision to schedule or defer RQ cases. An illustration of the decision tree associated with this scenario is presented in Figure 2. Of course, in practice, an OR manager must weigh these costs for multiple RQ cases across multiple ORs over the course of several days leading up to the day of surgery.

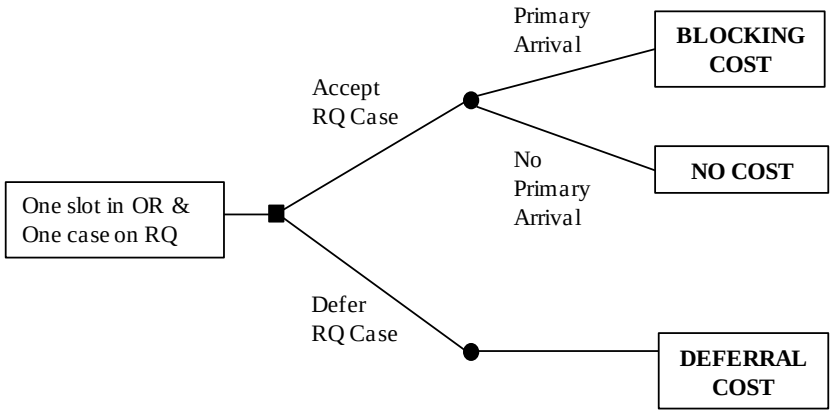

Figure 2. The OR manager's decision tree for a simple RQ scenario

These satisfaction costs highlight the often contentious nature of surgery scheduling decisions, and any attempt to study block release and RQ policies must take these factors into account. The primary motivation for the remainder of our research on the surgery scheduling problem, as presented in the next two chapters, is to mathematically model the interaction of these satisfaction costs with surgical demand patterns and explore their role in making optimal block release and RQ decisions. Rather than focus on traditional block scheduling and individual patient scheduling objectives (such as leveling hospital occupancy and maximizing OR utilization), our analysis will focus on how OR managers can make equitable scheduling decisions in the face of competing demands from various surgeons and surgical specialties.

In this chapter, combining a review of the existing literature with a case study of the scheduling system at UMMC, we have argued that the transition between the block scheduling and individual patient scheduling stages of surgery scheduling is the result of a dynamic interaction between the arrival patterns for both primary and secondary demand and the OR manager's RQ policies and block release decisions. In the following two chapters we will develop and study a stochastic dynamic programming (SDP) formulation of this sequential decision-making process. Because this approach to the surgery scheduling problem is new to the literature, our analysis will focus on capturing the most important relationships and developing a better understanding of how these relationships contribute to the evolution of the OR schedule.

Chapter 3. The Single-Day Surgery Scheduling Problem: General Formulation and a Special Case

Based on the structure of the surgery scheduling system at the University of Maryland Medical Center and the insights gained from our observation of the evolution of a single day's operating room schedule, we proceed with our formulation and analysis of the dynamic single-day surgery scheduling problem (SDSSP). The limited existing literature on the interaction of block schedules and patient scheduling suggests that the structure and insights from our case study are generalizable to other hospitals that use block scheduling (Dexter et al. 2003, Dexter and Macario 2004, Ozkarahan 2000). For this reason, our treatment of the SDSSP is aimed not at solving a single hospital's scheduling problem, but rather at developing general insights into how operating room managers can balance competing demand classes to make equitable request queue decisions.

3.1. Problem Statement and General Formulation

Problem Statement

As discussed above, we can limit the general problem of scheduling a suite of operating rooms to the specific problem that considers only one day's schedule. As a starting point, we assume that a suite of identical ORs has been allocated on the day in question to surgical specialties according to a block schedule. Demand for elective surgery is generated over time and enters the scheduling system as either primary demand or secondary demand, as described above. Primary demand is divided according to the primary specialties associated with each of the ORs, therefore we can think of each OR as having a separate source of primary demand. In contrast, secondary demand is not disaggregated in this way. Each of the demand sources is assumed to be stochastic, both in the quantity of demand generated and the timing of its arrival. When primary demand arrives to the system, it is added immediately to the schedule for its associated OR, providing there is adequate space. If there is not adequate space, then the excess primary demand is directed to the RQ. Secondary

demand is added directly to the RQ, where all RQ cases must wait for the OR manager to make RQ decisions.

Two types of costs are associated with the process of developing an OR schedule using a block scheduling system: (1) utilization costs associated with the ORs and (2) the deferral and blocking customer satisfaction costs defined at the end of Chapter 2. The utilization costs are common in the literature, but the customer satisfaction costs are a distinctive feature of this formulation. On each day leading up to the day of surgery, the OR manager must choose the number of RQ cases to add to each OR's schedule for the day of surgery. The manager's objective is to minimize the expected total cost of deferral and blocking penalties incurred on the days before surgery and the OR utilization costs on the day of surgery. In general, block release dates restrict the days on which RQ cases can be added to the schedule and may differ from room to room and RQ policies dictate which decisions to make in which scenarios.

The dynamic relationship between decisions and costs suggests a stochastic dynamic programming (SDP) formulation. Because block release dates serve as constraints on the OR manager's RQ decisions, it is clear that an optimal RQ policy combination would use no block release dates and the optimal decisions from the unconstrained SDP as the RQ policy. For this reason, the formulation and analysis in the subsequent sections consider an SDP unconstrained by block release dates. However, because block release dates are used in practice, the added cost of imposing them will be studied in Chapter 4.

General Formulation

We proceed with a SDP formulation for the general SDSSP. We assume that the demand sources (primary for each OR and secondary) are independent of each other, as are the number of arrivals from day to day for a given source. RQ decisions for each OR are made once a day on each day leading up to the day of surgery, and are made before any additional cases for the day of surgery arrive. For modeling purposes, cases will be separated into different types based on their scheduled durations (the amount of OR time allocated to the case). While this approach requires discretizing the case durations, which may not capture the entire range of possible

durations, there is a precedent in the literature for treating case durations in this manner (see, for instance, Guinet and Chaabane 2003). In addition to helping maintain the computational tractability of the resulting SDP, this approach allows the analysis to stay focused on the principal issue of how OR managers balance potential blocking and deferral costs.

The input data for the formulation is defined as follows:

$N =$ number of days before the day of surgery on which surgical demand is generated

$S =$ number of identical rooms in the OR suite

$C =$ capacity of the ORs (in hours)

$K =$ number of case types

$d_k =$ duration (in hours) of cases of type k, for $k = 1, \dots, K$

The stochastic demand for surgery is given by two classes of random variables associated with primary cases and secondary cases. From this point forward, all references to days refer to the number of days before the day of surgery, with day 0 representing the day of surgery. For $j = 1, \dots, N$, $k = 1, \dots, K$, and $s = 1, \dots, S$:

$T_j^{sk} =$ number of primary cases of type k that arrive to room s on day j

$R_j^k =$ number of secondary cases of type k that arrive on day j

The blocking and deferral costs for each day before surgery and the utilization costs for the day of surgery are similarly defined. For $k = 1, \dots, K$ and $s = 1, \dots, S$:

$h_0^k =$ penalty for unscheduled cases of type k left on RQ on the day of surgery

$r_0^s =$ penalty for unused space in room s on the day of surgery

For $j = 1, \dots, N$, $k = 1, \dots, K$, and $s = 1, \dots, S$:

$h_j^k =$ deferral cost on day j for cases of type k

$r_j^{sk} =$ blocking cost on day j for cases of type k arriving to room s

Three classes of state variables are required to represent the state of each OR's schedule at the start of day j (before day j's RQ decisions have been made and before any new cases have arrived). Naturally, states are required to keep track of the number of available hours remaining in each room and the number of cases of each type on the RQ. The extra state variable, the number of blocking eligible hours in each OR, reflects the presence of RQ cases that were added to the schedule on previous days but have not yet incurred a blocking penalty. This auxiliary state accounts for the fact that a primary case might be blocked by a RQ case scheduled on a previous day. For $j = 0, \dots, N$, $k = 1, \dots, K$, and $s = 1, \dots, S$:

C_j^s = available hours remaining on room s's schedule on day j

B_j^s = number of blocking eligible hours on room s's schedule on day j

W_j^k = number of cases of type k on the RQ on day j

Finally, the decision variables can be defined. For $j = 0, \dots, N$, $k = 1, \dots, K$, and $s = 1, \dots, S$:

x_j^{sk} = number of RQ cases of type k to add to room s's schedule on day j

With the exception of the day of surgery, the costs incurred each day are separated into deferral costs and blocking costs. Deferral penalties are only assessed up to the number of RQ placements that are feasible, and can be computed with the following expression. For $j = 1, \dots, N$ and $k = 1, \dots, K$:

$$ND_j^k = \text{number of cases of type } k \text{ deferred on day } j$$
$$= \min\left(\sum_{s=1}^{S}\left\lfloor\frac{C_j^s}{d_k}\right\rfloor, W_j^k\right) - \sum_{s=1}^{S} x_j^{sk}$$

When the RQ decisions are made on day j, that day's arrivals have not yet occurred. Therefore, the number of blocking penalties incurred on day j is a random variable that depends upon the decision variables and on the arrival of primary cases. The consideration of different case durations raises an important question related to computing blocking costs. If a longer primary case is blocked and sent to the RQ,

should a blocking penalty be assessed for the entire case or only for the part of the case that overlaps with the blocking eligible hours on the OR schedule? Suppose, for example, that the schedule for room s has one available hour ($C_j^s = 1$) and one blocking eligible hour ($B_j^s = 1$) and that a primary case with a duration of two hours arrives ($T_j^{sk} = 1$ for the case type with $d_k = 2$). Because this case will not fit in the available time, it is blocked and placed on the RQ. The blocking penalty can be assessed either for the entire two hour duration or for just the fraction of the case that overlaps with the blocking eligible hour. (Both approaches will be considered during computational testing in Chapter 4). These blocking quantities cannot be computed with a closed form expression, but instead require an algorithm. The pseudocode for this algorithm, which we refer to as *ProcessDay*, is presented later in this subsection. The algorithm returns the following values, which are important for computing blocking costs and for writing the transition and optimality equations. For $j = 1, \dots, N$, $k = 1, \dots, K$, and $s = 1, \dots, S$:

NB_j^{sk} = number of cases of type k blocked on day j after arriving to room s

FB_j^{sk} = fraction of cases of type k blocked on day j after arriving to room s

fit_j^{sk} = number of primary cases of type k scheduled in room s on day j

This is now sufficient information to define the value function, Bellman's optimality equation, and the day-to-day transition equations. Note that the daily blocking penalty for each room is a convex combination of the number blocked and the fraction blocked, with the weighting parameter λ satisfying $0 \leq \lambda \leq 1$. This parameter allows us to address the question of full or partial blocking, and its impact will be explored during computational testing. For convenience, some quantities are expressed in vector form (across operating rooms, case types, or both) in the optimality equation and subsequent discussion. For $j = 1, \dots, N$:

$$V_j(\boldsymbol{C_j}, \boldsymbol{B_j}, \boldsymbol{W_j}) = \text{minimum expected remaining cost from state } (\boldsymbol{C_j}, \boldsymbol{B_j}, \boldsymbol{W_j}) \text{ on day } j$$

$$= \min_{\boldsymbol{x_j}} \left\{ \sum_{k=1}^{K} \left(h_j^k ND_j^k + \sum_{s=1}^{S} r_j^{sk} \, E\left[\lambda \, NB_j^{sk} + (1-\lambda) FB_j^{sk}\right] \right) + E\left[V_{j-1}(\boldsymbol{C_{j-1}}, \boldsymbol{B_{j-1}}, \boldsymbol{W_{j-1}})\right] \right\}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} d_k x_j^{sk} \le C_j^s \quad \text{for } s = 1, \dots, S$$

$$0 \le \sum_{s=1}^{S} x_j^{sk} \le W_j^k \quad \text{for } k = 1, \dots, K$$

$$x_j^{sk} \text{ integer}$$

where, for $s = 1, \dots, S$ and $k = 1, \dots, K$:

$$C_{j-1}^s = C_j^s - \sum_{k=1}^{K} d_k (x_j^{sk} + fit_j^{sk})$$

$$B_{j-1}^s = B_j^s + \sum_{k=1}^{K} d_k (x_j^{sk} - FB_j^{sk})$$

$$W_{j-1}^k = W_j^k + R_j^k - \sum_{s=1}^{S} x_j^{sk} - (T_j^{sk} - fit_j^{sk})$$

The boundaries for the SDP are day N and day 0. On day N the system is initialized to empty, and on day 0 the boundary costs come from the unused time on each of the OR's schedules and the unscheduled cases that remain on the RQ. We note that deferral and blocking penalties are not a concern on the day of surgery, because we assume that all elective demand arrives to the system prior to day 0's RQ decisions. As with the daily deferral costs, the objective function on day 0 only penalizes those cases left on the RQ that could have been feasibly added to the schedule. The boundary value function is defined as follows:

$$V_0(\boldsymbol{C_0}, \boldsymbol{B_0}, \boldsymbol{W_0}) = \min_{\boldsymbol{x_0}} \left\{ \sum_{k=1}^{K} h_0^k \left(\min \left(\sum_{s=1}^{S} \left\lfloor \frac{C_0^s}{d_k} \right\rfloor, W_0^k \right) - \sum_{s=1}^{S} x_0^{sk} \right) + \sum_{s=1}^{S} r_0^s \left(C_0^s - \sum_{k=1}^{K} d_k x_0^{sk} \right) \right\}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} d_k x_0^{sk} \leq C_0^s \text{ for } s = 1, \dots, S$$

$$0 \leq \sum_{s=1}^{S} x_0^{sk} \leq W_0^k \text{ for } k = 1, \dots, K$$

$$x_0^{sk} \text{ integer}$$

This day 0 value function is an integer program with some additional constants in the objective function, and for a single operating room ($S = 1$) it reduces to an integer knapsack problem.

The final details needed to complete the formulation involve some aspects of the *ProcessDay* algorithm for computing the blocking quantities for each room. The algorithm represents how the system processes a particular realization of the primary demand arrival random variables for each of the operating rooms. The pseudocode for this algorithm is presented below. In order to process primary arrivals of different types, it is necessary to assume some sort of case type prioritization on behalf of the primary specialty. Recall from Chapter 2 that this prioritization is beyond the control of the OR manager, so we do not include it as a decision variable in the model. (Note that prioritization between the RQ case types is within the OR manager's control and is reflected as such in the daily RQ decision variables). We assume throughout that the primary specialty prioritizes its cases by decreasing duration. That is, longer duration cases receive priority over the shorter duration cases. Other prioritization rules could easily be considered within the framework of the algorithm.

Pseudocode for *ProcessDay* Algorithm

INPUT: $C_j^s, B_j^s, x_j^{sk}, T_j^{sk}, d_k$ for $k = 1, \dots, K$ and a given room s

DEFINE:

fit_j^{sk} = number of primary arrivals of type k placed into room s on day j

$blfit_j^{sk}$ = number of primary arrivals of type k that would have fit in room s on day j if the blocking eligible hours were available

INITIALIZE:

$\tilde{C}_j^s = C_j^s - \sum_{k=1}^{K} d_k x_j^{sk}$ // space available in room after RQ decisions

$\tilde{B}_j^s = B_j^s + \sum_{k=1}^{K} d_k x_j^{sk}$ // blocking eligible hours after RQ decisions

$\tilde{D}_j^s = \tilde{C}_j^s + \tilde{B}_j^s$ // space available if blocking hours were free

FOR $k = K$ TO 1: // reflects prioritization of primary arrivals

// determine cases of type k that either fit or would have fit into open space

$tmpfit = \frac{\tilde{C}_j^s}{d_k}$

$fit_j^{sk} = min\left(T_j^{sk}, \lfloor tmpfit \rfloor\right)$

$blfit_j^{sk} = min\left(T_j^{sk}, \left\lfloor \frac{\tilde{D}_j^s}{d_k} \right\rfloor\right)$

// compute blocking penalties

$NB_j^{sk} = blfit_j^{sk} - fit_j^{sk}$

$FB_j^{sk} = \max\left(0, blfit_j^{sk} - tmpfit\right)$

// compute remaining space for next iteration

$\tilde{C}_j^s = \tilde{C}_j^s - d_k\, fit_j^{sk}$

$\tilde{D}_j^s = \tilde{D}_j^s - d_k\, blfit_j^{sk}$

END-FOR

RETURN: $fit_j^{sk}, NB_j^{sk}, FB_j^{sk}$ for $k = 1, \dots, K$ and a given room s

We conclude our presentation of the general formulation for the SDSSP with a comment about the computational complexity involved in solving this initial SDP. As with many SDPs, this formulation is susceptible to the curse of dimensionality. The number of feasible states for a given operating room (not taking into account the RQ)

is linear in the number of days before surgery (N) and quadratic in the capacity of the OR (due to possible combinations of C_j^s and B_j^s). Combining these states across operating rooms already yields a combined complexity that is exponential in the number of rooms. While there is no theoretical bound on the size of the RQ, it is clear that any practical truncation must maintain enough space for each case type to fill the available capacity in each of the rooms (we will return to this truncation issue later in Chapter 4). This introduces additional complexity that is exponential in the number of case types. Without even considering the range of feasible decision values or the possible supports for the arrival random variables, we see that the SDP for the general SDSSP quickly becomes intractable for large problems. The case study in Chapter 2 shows that realistic values of N can exceed 20 days (with non-trivial decisions typically occurring in the final 5 days) and values for the OR capacity (C) are typically eight hours. At UMMC, there are over 20 ORs and case durations can range from less than an hour to over eight hours. We tackle this complexity problem by first analyzing a special case of SDSSP with one operating room and one case type. The results of this analysis will motivate heuristic approaches for SDSSP that we will present and test in Chapter 4.

3.2. A Single OR with Unit Durations

Because of the novelty of this approach to the single-day surgery scheduling problem, we begin with an analysis of the simplest case, a single operating room where all cases have the same (unit) duration. As the following discussion will show, the optimal daily solutions for this special case, which we will refer to as SDSSP1, follow a threshold pattern that reserves a specific amount of space in the OR for the remaining primary demand on each day leading up the day of surgery. This intuitive result, together with the algorithm for determining the optimal thresholds, serves as the basis for our continued analysis of more general cases with multiple case types and multiple operating rooms in Chapter 4.

Simplified Formulation

The general formulation simplifies considerably for SDSSP1. While we refrain from restating the entire simplified formulation, several pieces of the simplification are helpful in proving that the optimal solutions follow the threshold pattern mentioned above. Throughout our discussion of SDSSP1, we will drop the operating room and case type indices (s and k, respectively) from our notation. The definitions of the arrival random variables (T_j and R_j), daily cost parameters (h_j and r_j), state variables (C_j, B_j, and W_j), and decision variables (x_j) require no further modification.

The expression for the number of deferral and the *ProcessDay* algorithm for the number of blocking penalties incurred on day j both simplify considerably. First, the number of deferral penalties incurred on day j simplifies to the following:

$$ND_j = \min(W_j, C_j) - x_{sj}$$

Because SDSSP1 does not consider different case types, the number of blocking penalties can actually be computed using a closed expression, rather than requiring the *ProcessDay* algorithm. Recall that the number of blocking penalties depends on the realization of the primary arrivals. Figure 3 illustrates how the blocking penalties change between three different demand scenarios, which leads to the following expression for the number of blocking penalties on day j.

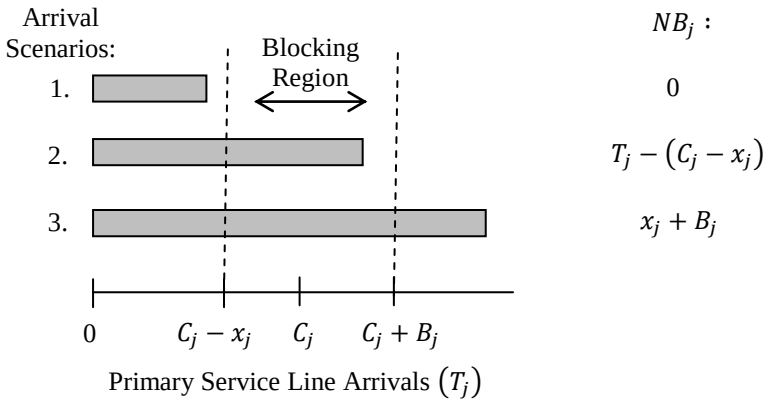

Figure 3. Illustration of different blocking penalty scenarios for SDSSP1

$$NB_j = \begin{cases} 0 & \text{if } T_j \le C_j - x_j \\ B_j + x_j & \text{if } T_j \ge C_j + B_j \\ T_j - (C_j - x_j) & \text{otherwise} \end{cases}$$
$$= \max\big(0, \min\big(B_j + x_j, T_j - C_j + x_j\big)\big)$$

The figure and the resulting expression show that if the primary arrivals will fit into the available space in the room (after that day's RQ decision), then no blocking penalty is incurred. On the other hand, if the number of arrivals is large, the number of blocking penalties is capped by the number of blocking eligible cases in the room (from previous days' and from that day's RQ decisions).

The transitions from day j to day $j-1$ can also be simplified considerably using this new expression for day j's blocking penalties.

$$C_{j-1} = C_j - x_j - \min\big(T_j, C_j - x_j\big)$$
$$B_{j-1} = B_j + x_j - NB_j$$
$$W_{j-1} = W_j - x_j + R_j + \max\big(0, T_j - C_j + x_j\big)$$

Finally, in order to facilitate our analysis of the value function minimization problem, we introduce the following notation referring to the function that each day's value function aims to minimize. First, for $j = 1, \dots, N$, and then, for the day of surgery ($j = 0$).

$$F_j\big(C_j, B_j, W_j, x_j\big) = h_j ND_j + r_j E\big[NB_j\big] + E\big[V_{j-1}\big(C_{j-1}, B_{j-1}, W_{j-1}\big)\big]$$

$$F_0(C_0, B_0, W_0, x_0) = h_0(\min\,(C_0, W_0) - x_0) + r_0(C_0 - x_0)$$

This allows us to restate the value function for all j:

$$V_j\big(W_j, B_j, C_j\big) = \min\{F_j\big(C_j, B_j, W_j, x_j\big) \mid x_j \in \{0,1, \dots, \min\big(W_j, C_j\big)\}\}$$

This notation also allows us to introduce a notation for the optimal SDP decision corresponding to state $\big(C_j, B_j, W_j\big)$. We also define a finite difference function for the function $F_j\big(C_j, B_j, W_j, x_j\big)$ with respect to x_j, which we will require later in our analysis.

$$x_j(C_j, B_j, W_j) = \arg\min\{F_j(C_j, B_j, W_j, x_j) \mid x_j \in \{0,1, \dots, \min(W_j, C_j)\}\}$$
$$\Delta F_j(C_j, B_j, W_j, x_j) = F_j(C_j, B_j, W_j, x_j + 1) - F_j(C_j, B_j, W_j, x_j)$$

The following two subsections study the optimal decisions for SDSSP1. We first discuss a sample path for a simple example and observe the threshold behavior that emerges from the policy maps for the optimal decisions. We then formalize this threshold behavior and prove both its existence and optimality.

Optimal Solution Patterns

In order to better understand the optimal decisions generated by the SDP, and how they translate to blocking and deferral costs, it is helpful to look at a sample path for a small numerical example. Suppose an OR has capacity for four cases, and that the daily primary and secondary demand arrivals both follow Poisson distributions. The daily arrival rates and system costs are specified in Table 8. The resulting SDP is solved to get the optimal decision for each feasible state on each day. Table 9 illustrates the sample path generated by single realizations of the arrival random variables, combined with the optimal decisions and transitions generated by the SDP.

Of particular interest in this example are days 3 and 2, where the optimal SDP decisions for the given states say to schedule one case off the RQ on each day. On day 3, the RQ consists of two cases but only one is taken, leading to a deferral penalty. Because there is still sufficient space in the room for the primary service line arrival ($T_3 < C_3 - x_3$), no blocking penalty is incurred and one blocking eligible case is passed to the following day. On day 2, another deferral penalty is incurred, and because the primary service line's arrivals exceed the remaining available space ($T_2 > C_2 - x_2$) a blocking penalty is also incurred. On the day of surgery, we see that there is no available time in the OR ($C_0 = 0$), while there are still three cases on the RQ. Because penalties for not scheduling RQ cases are only assessed up to what is feasible, and no placements are feasible on day 0, no costs are incurred for these unscheduled cases.

Table 8. Input data for a simple SDSSP1 instance

	Days Before Surgery				
	4	3	2	1	0
Arrival Rates					
Primary	1	2	0.5	0.5	0
Secondary	1	1	1	1	0
Costs					
Deferral (h_j)	1	1	1	1	1
Blocking (r_j)	3	3	3	3	5

Note: Capacity of OR is set to 4 cases and random variables follow Poisson distributions.

Table 9. Sample path for a simple SDSSP1 instance using the optimal SDP solutions

	Days Before Surgery (j)				
	4	3	2	1	0
W_j	0	2	2	2	3
B_j	0	0	1	1	0
C_j	4	4	2	0	0
x_j	0	1	1	0	0
T_j	0	1	2	1	0
R_j	2	1	0	0	0
ND_j	0	1	1	0	3
NB_j	0	0	1	1	0
Daily Costs	0	1	4	3	0

In the exploration of optimal policies for SDSSP1 across a wide range of input data, a striking trend emerges. For all input data satisfying certain realistic assumptions, the optimal policy for each day follows what we describe as a threshold policy. That is, for each day before surgery there is a specific amount of space preserved for future primary arrivals, and the optimal decision takes as many cases as necessary to reach this threshold. If this number of cases is not feasible, then the decision takes the system as close to the threshold as possible. Furthermore, the threshold is independent of the RQ demand arrival process and the number of blocking eligible cases already in the OR. For the example above, the observed thresholds were (2, 3, 1, 1, 0) for days 4,…,0. Looking at these thresholds for the states observed in the sample path in Table 9 sheds light on why the corresponding decisions are made (e.g., on day 3, four spaces are available and the threshold is three,

so one RQ case is scheduled). Figure 4 shows how the optimal decisions change as the available time in the OR increases for a hypothetical day on which the threshold says to preserve time for two future primary cases. If the available time is already less than the threshold, then no RQ cases are scheduled. On the other extreme, if there are not enough cases on the RQ to reach the threshold, then the optimal decision takes as many RQ cases as are present.

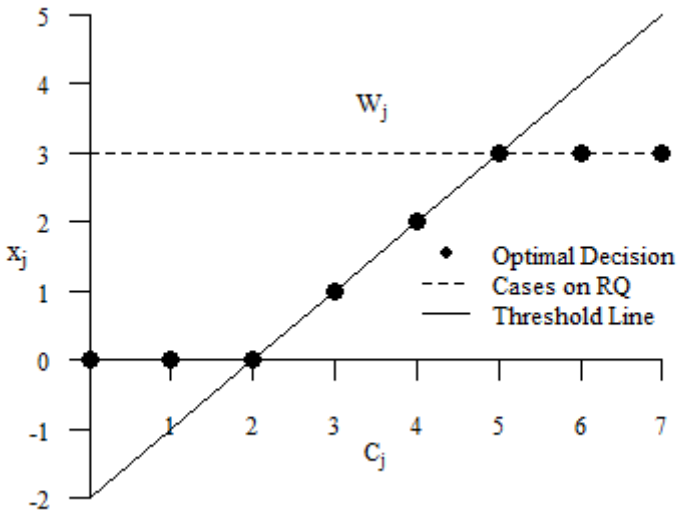

Figure 4. Sample policy map showing threshold behavior for SDSSP1

The next section presents an analytical proof that the optimal policies for the single room SDP always demonstrate this threshold behavior. The proof leads to a constructive algorithm for finding the desired thresholds, and thus all optimal decisions, for any set of input data without solving the full SDP.

Optimality of the Threshold Behavior

In the course of an induction proof for the optimality of the threshold behavior, the nature of the costs and state transitions for certain adjacent states and decision values will be important. The relationships below for $j = 1, \ldots, N$ result from the transition equations presented with the SDP formulation above. If explicit dependence on the state variables, decision variables, or arrival random variables is not shown, then these values are held constant in the differences below. The notation

$I\{\ldots\}$ represents an indicator random variable, and the relationships are grouped for clarity.

<u>*Group 1*</u>: $NB_j(x_j+1) - NB_j(x_j) = I\{T_j \geq C_j - x_j\}$

$C_{j-1}(x_j+1) - C_{j-1}(x_j) = -I\{T_j < C_j - x_j\}$

$B_{j-1}(x_j+1) - B_{j-1}(x_j) = I\{T_j < C_j - x_j\}$

$W_{j-1}(x_j+1) - W_{j-1}(x_j) = -I\{T_j < C_j - x_j\}$

<u>*Group 2*</u>: $NB_j(C_j-1, B_j+1) - NB_j(C_j, B_j) = I\{T_j \geq C_j - x_j\}$

$C_{j-1}(C_j-1) - C_{j-1}(C_j) = -I\{T_j < C_j - x_j\}$

$B_{j-1}(C_j-1, B_j+1) - B_{j-1}(C_j, B_j) = I\{T_j < C_j - x_j\}$

$W_{j-1}(C_j-1, W_j-1) - W_{j-1}(C_j, W_j) = -I\{T_j < C_j - x_j\}$

<u>*Group 3*</u>: $N_j^b(B_j+1, C_j-1, x_j-1) = N_j^b(B_j, C_j, x_j)$

$C_{j-1}(C_j-1, x_j-1) = C_{j-1}(C_j, x_j)$

$B_{j-1}(C_j-1, B_j+1, x_j-1) = B_{j-1}(C_j, B_j, x_j)$

$W_{j-1}(C_j-1, W_j-1, x_j-1) = W_{j-1}(C_j, W_j, x_j)$

These relationships provide the necessary insights to proceed with a formal statement and proof of the threshold policy suggested above. Two realistic assumptions on the input data are required: (1) $h_j \leq r_j$, $\forall j$ and (2) $r_{j+1} \geq r_j$, $\forall j \geq 1$. These do not limit the strength of the result, because (1) deferral costs are certain while blocking penalties depend on uncertain future arrivals and (2) increasing the blocking penalty as the day of surgery approaches would discourage filling up the remaining space. In the statement of the theorem, part (iii) is the desired threshold result, while the other parts are necessary in the development of the proof.

Theorem 3.1 *For* $j = N, \ldots, 1$ *and all feasible states* (C_j, B_j, W_j),

(i) $\exists$ *a function* $G_j(n)$ *s.t.* $G_j(n)$ *is non-increasing in* n, $G_j(1) \leq r_j$ *and* $\Delta F_j(C_j, B_j, W_j, x_j) = G_j(C_j - x_j)$

(ii) $F_j(C_j, B_j, W_j, x_j)$ *is convex in* x_j

(iii) $\exists\, Y_j$ *satisfying* $G_j(Y_j + 1) < 0 \leq G_j(Y_j)$ *s.t.*

$$x_j(C_j, B_j, W_j) = min(W_j, max(0, C_j - Y_j))$$

(iv) $V_j(C_j - 1, B_j + 1, W_j - 1) - V_j(C_j, B_j, W_j) = \begin{cases} 0 & if\ C_j - Y_j > 0 \\ G_j(C_j) & if\ C_j - Y_j \leq 0 \end{cases}$

The proof proceeds using weak induction.

Base Case ($j = 1$): Observe the following statements about day 0. First, no OR slots need to be preserved for future arrivals, leading to an effective threshold of $K_0 = 0$. This gives the desired structure to the optimal decisions.

$$\begin{aligned} x_0(C_0, B_0, W_0) &= \min(W_0, \max(0, C_0 - K_0)) \\ &= \min(W_0, C_0) \end{aligned}$$

Second, substituting this choice of x_0 into the day 0 value function, followed by some simplification, allows the value function to be expressed in terms of $(W_0 - C_0)$. This observation leads to the following equality for day 0.

$$V_0(C_0 - 1, B_0 + 1, W_0 - 1) = V_0(C_0, B_0, W_0)$$

Using the Group 1 transition relationships and this day 0 equality, the first two desired statements for day 1 emerge:

$$\begin{aligned} \Delta F_1(W_1, B_1, C_1, x_1) &= F_1(W_1, B_1, C_1, x_1 + 1) - F_1(W_1, B_1, C_1, x_1) \\ &= h_1[ND_1(x_1 + 1) - ND_1(x_1)] + r_1[NB_1(x_1 + 1) - NB_1(x_1)] \\ &\quad + E\big[V_0\big(C_0(x_1 + 1), B_0(x_1 + 1), W_0(x_1 + 1)\big) \\ &\qquad - V_0\big(C_0(x_1), B_0(x_1), W_0(x_1)\big)\big] \\ &= -h_1 + r_1 E[I\{T_1 \geq C_1 - x_1\}] \\ &= -h_1 + r_1 P[T_1 \geq C_1 - x_1] \end{aligned}$$

$$= G_1(C_1 - x_1)$$

where $G_1(n) = -h_1 + r_1 P[T_1 \geq n]$

Note that $G_1(1) \leq -h_1 + r_1 \leq r_1$. Also, $G_1(n)$ is clearly non-increasing in n, which gives $\Delta F_1(C_1, B_1, W_1, x_1)$ non-decreasing in x_1 and proves the convexity of $F_1(C_1, B_1, W_1, x_1)$ with respect to x_1.

Minimizing $F_1(C_1, B_1, W_1, x_1)$ then suggests looking for the point where the finite difference changes signs. In other words, seek out Y_1 such that $G_1(Y_1 + 1) < 0 \leq G_1(Y_1)$ and try to set $x_1 = C_1 - Y_1$. Such a Y_1 exists because $G_1(0) = -h_1 + r_1 \geq 0$ (by assumption on input data) and because $G_1(n) \to -h_1 < 0$ as $n \to \infty$. If the desired value for x_1 is infeasible, then choose x_1 on the boundary closest to the desired value. This gives the desired expression for the optimal decision in terms of the threshold Y_1.

$$x_1(C_1, B_1, W_1) = \min(W_1, \max(0, C_1 - Y_1))$$

All that remains for the base case is to show part (iv) of the claim, which is critical for the inductive step. For brevity, define $x_1^{**} = x_1(C_1 - 1, B_1 + 1, W_1 - 1)$ and $x_1^* = x_1(C_1, B_1, W_1)$. From the expression above for the optimal decisions, there are two cases to consider: (1) $x_1^{**} = x_1^* = 0$ when $C_1 - Y_1 \leq 0$ and (2) $x_1^{**} = x_1^* - 1$ when $C_1 - Y_1 > 0$.

Case 1 ($x_1^{**} = x_1^* = 0$): The group 2 transition relationships state that in this scenario the subsequent day 0 states (from the corresponding x_1^{**} and x_1^* day 1 states) will either be identical (for $T_1 \geq C_1$) or differ in such a way that by the day 0 equality above they will have the same value (for $T_1 < C_1$). The desired difference then depends only on the differences in deferral and blocking costs.

$$\begin{aligned} V_1(C_1 - 1, B_1 + 1, W_1 - 1) - V_1(C_1, B_1, W_1) & \\ &= h_1[ND_1(C_1 - 1, W_1 - 1) - ND_1(C_1, W_1)] \\ &\quad + r_1 E[NB_1(C_1 - 1, B_1 + 1) - NB_1(C_1, B_1)] \\ &= -h_1 + r_1 \cdot P[T_1 \geq C_1] \\ &= G_1(C_1) \end{aligned}$$

Case 2 ($x_1^{**} = x_1^* - 1$): According to the group 3 transition relationships, the blocking costs and subsequent day 0 states will be identical in this scenario. The deferral costs will also be the same, giving:

$$V_1(W_1 - 1, B_1 + 1, C_1 - 1) - V_1(W_1, B_1, C_1) = 0$$

These two cases yield the final piece of the base case.

Inductive Step: Assume that all parts of the claim hold for day j–1.

$$\begin{aligned}\Delta F_j(C_j, B_j, W_j, x_j) &= F_j(C_j, B_j, W_j, x_j + 1) - F_j(C_j, B_j, W_j, x_j) \\ &= h_j[ND_j(x_j + 1) - ND_j(x_j)] + r_j E[NB_j(x_j + 1) - NB_j(x_j)] \\ &\quad + E\left[V_{j-1}\left(C_{j-1}(x_j + 1), B_{j-1}(x_j + 1), W_{j-1}(x_j + 1)\right)\right. \\ &\quad \left. - V_{j-1}\left(C_{j-1}(x_j), B_{j-1}(x_j), W_{j-1}(x_j)\right)\right]\end{aligned}$$

By the group 1 relationships, the day j–1 states are identical when $T_j \geq C_j - x_j$. Otherwise, the states have the form of the difference in part (iv) of the induction assumption. By the induction assumption then, the resulting values are equal when $C_{j-1}(x_j) - K_{j-1} > 0$. But for $T_j < C_j - x_j$, it follows that $C_{j-1}(x_j) = C_j - x_j - T_j$. Combining these pieces with the induction assumption, the difference in day j–1 states is $G_{j-1}(C_j - x_j - T_j)$ when $C_j - x_j - K_{j-1} \leq T_j < C_j - x_j$, and is zero otherwise. This is reflected in the conditional expectation below, allowing the finite difference calculation to continue.

$$\begin{aligned}\Delta F_j(C_j, B_j, W_j, x_j) &= -h_j + r_j P[T_j \geq C_j - x_j] \\ &\quad + E[G_{j-1}(C_j - x_j - T_j) | C_j - x_j - Y_{j-1} \leq T_j < C_j - x_j] \\ &= -h_j + r_j P[T_j \geq C_j - x_j] + \sum_{i=1}^{Y_{j-1}} P[T_j = C_j - x_j - i] G_{j-1}(i) \\ &= G_j(C_j - x_j)\end{aligned}$$

where $G_j(n) = -h_j + r_j P[T_j \geq n] + \sum_{i=1}^{Y_{j-1}} P[T_j = n - i] G_{j-1}(i)$

It is important to note that if $Y_{j-1} = 0$, then the final summation in this expression disappears. Moving on to show that $G_j(n)$ possesses the desired qualities, the next two results use both the induction assumptions on $G_{j-1}(n)$ and the assumption that $r_j \geq r_{j-1}$.

$$\begin{aligned}
G_j(1) &= -h_j + r_j P\big[T_j \geq 1\big] + \sum_{i=1}^{Y_{j-1}} P\big[T_j = 1 - i\big] G_{j-1}(i) \\
&= -h_j + r_j P\big[T_j \geq 1\big] + P\big[T_j = 0\big] G_{j-1}(1) \\
&\leq -h_j + r_j P\big[T_j \geq 1\big] + P\big[T_j = 0\big] r_{j-1} \\
&\leq -h_j + r_j \\
&\leq r_j
\end{aligned}$$

$$\begin{aligned}
&G_j(n-1) - G_j(n) \\
&\quad = \left\{ -h_j + r_j P\big[T_j \geq n-1\big] + \sum_{i=1}^{Y_{j-1}} P\big[T_j = n-1-i\big] G_{j-1}(i) \right\} \\
&\qquad - \left\{ -h_j + r_j P\big[T_j \geq n\big] + \sum_{i=1}^{Y_{j-1}} P\big[T_j = n-i\big] G_{j-1}(i) \right\} \\
&\quad = r_j P\big[T_j = n-1\big] + P\big[T_j = n-1-Y_{j-1}\big] G_{j-1}\big(Y_{j-1}\big) \\
&\qquad + \sum_{i=1}^{Y_{j-1}-1} P\big[T_j = n-i\big] \Big(G_{j-1}(i) - G_{j-1}(i+1) \Big) \\
&\qquad - P\big[T_j = n-1\big] G_{j-1}(1)
\end{aligned}$$

At this stage, note that $G_{j-1}\big(Y_{j-1}\big) \geq 0$ by the selection of Y_{j-1} and $G_{j-1}(i) - G_{j-1}(i+1) \geq 0$ and $G_{j-1}(1) \leq r_{j-1}$ by the induction assumption. Continuing with the computation gives:

$$G_j(n-1) - G_j(n) \geq r_j P\big[T_j = n-1\big] - r_{j-1} P\big[T_j = n-1\big] \geq 0$$

Therefore $G_j(n)$ is non-increasing in n, which gives $F_j\big(C_j, B_j, W_j, x_j\big)$ convex in x_j. Using the same argument presented in the base case, minimizing $F_j\big(C_j, B_j, W_j, x_j\big)$

requires finding Y_j such that $G_j(Y_j + 1) < 0 \leq G_j(Y_j)$ and trying to set $x_j = C_j - Y_j$. Again, this Y_j is guaranteed to exist because $G_j(0) = -h_j + r_j \geq 0$ and because $G_j(n) \to -h_j < 0$ as $n \to \infty$. If the desired value for x_j is infeasible, then choose x_j on the boundary closest to the desired value. This gives the required expression for the optimal decision in terms of the threshold K_j:

$$x_j(C_j, B_j, W_j) = \min(W_j, \max(0, C_j - Y_j))$$

In order to finish up the final part of the claim, define $x_j^{**} = x_j(C_j - 1, B_j + 1, W_j - 1)$ and $x_j^* = x_j(C_j, B_j, W_j)$. Just as in the base case, the expression for the optimal decisions yields two cases for the relationship between these two policies: (1) $x_j^{**} = x_j^* = 0$ when $C_j - Y_j \leq 0$ and (2) $x_j^{**} = x_j^* - 1$ when $C_j - Y_j > 0$.

Case 1 ($x_j^{**} = x_j^* = 0$): The group 2 transition relationships state that in this scenario the subsequent day j–1 states (from the corresponding x_j^{**} and x_j^* states) will either be identical (for $T_j \geq C_j$) or differ in such a way that part (iv) of the induction assumption may be applied (for $T_j < C_j$). Applying the induction assumption when $T_j < C_j$ gives that the difference in values between the day j–1 states will be $G_{j-1}(C_{j-1})$ when $C_{j-1} - Y_{j-1} \leq 0$, and will be zero otherwise. But when $T_j < C_j$, note that $C_{j-1}(0) = C_j - T_j$. This implies that the value of the day j–1 states will only be nonzero when $C_j - Y_{j-1} \leq T_j < C_j$, a result which appears in the conditional expectation below.

$$\begin{aligned}
V_j(C_j - 1, B_j + 1, W_j - 1) - V_j(C_j, B_j, W_j) \\
&= h_j[ND_j(C_j - 1, W_j - 1) - ND_j(C_j, W_j)] \\
&\quad + r_j[NB_j(C_j - 1, B_j + 1) - NB_j(C_j, B_j)] \\
&\quad + E[G_{j-1}(C_j - T_j) \mid C_j - Y_{j-1} \leq T_j < C_j] \\
&= -h_j + r_j P[T_j \geq C_j] + \sum_{i=1}^{Y_{j-1}} P[T_j = C_j - i] G_{j-1}(i) \\
&= G_{j-1}(C_{j-1})
\end{aligned}$$

Case 2 ($x_j^{**} = x_j^* - 1$): As in the base case, the group 3 transition relationships show that the blocking costs and subsequent day $j-1$ states will be identical in this scenario. The deferral costs will also be the same, giving:

$$V_j(W_j - 1, B_j + 1, C_j - 1) - V_j(W_j, B_j, C_j) = 0$$

These cases complete the proof of the claim. ■

The definition of $G_j(n)$ and part (iii) of the claim suggest a constructive algorithm to determine the optimal thresholds for any set of input data. The proof of the base case demonstrates that in addition to setting $Y_0 = 0$, Y_1 can be found by iterating through $G_1(n)$ for $n = 0, 1, 2,$ ... until it changes sign. Using Y_{j-1} and storing $G_{j-1}(n)$ for $n = 1, 2, \ldots, Y_{j-1}$, $G_j(n)$ can similarly be computed and Y_j selected at the point where $G_j(n)$ changes sign.

Equipped with this algorithm for computing the optimal policy thresholds, optimal solutions for SDSSP1 can be obtained for any set of input data outside of the SDP framework (and thus escaping the curse of dimensionality associated with SDPs). In order to get a better feel for the sensitivity of the optimal thresholds, and thus the optimal RQ decisions, to the input data, the next subsection applies the optimal threshold algorithm to a range of input data

Computational Results

In the algorithm for computing the optimal policy thresholds, it is interesting to note that the room capacity and day 0 utilization costs play no role in determining the thresholds. In fact, what drives the thresholds are the primary arrival distributions (T_j) and the ratio of blocking (r_j) to deferral (h_j) costs. To demonstrate the sensitivity of the thresholds to these input parameters, we test a range of ratios on different arrival scenarios. Three arrival scenarios are considered, corresponding to early, middle, and late demand arrival patterns, and the daily arrivals are assumed to follow Poisson distributions with the rates shown in Table 10.

Table 10. Primary demand arrival rates for three arrival scenarios

Arrival Scenarios	Days Before Surgery 4	3	2	1	0
Early	2	1	0.5	0.5	0
Middle	1	1	1	1	0
Late	0.5	0.5	1	2	0

Note: Arrival random variables follow a Poisson distribution.

For each of these scenarios, the day 0 costs are set to 0, the deferral costs are fixed at 1, and a range of blocking costs is selected. For the purposes of this initial sensitivity testing, the blocking costs remain constant from day to day within each scenario. The optimal daily thresholds are computed using the algorithm suggested in the proof of Theorem 3.1. The resulting thresholds are shown in Table 11. In each scenario, a blocking-to-deferral cost ratio of 1:1 leads to thresholds of 0, in effect equivalent to a greedy RQ policy. For larger ratios, the threshold patterns mirror the arrival patterns. Ratios in the range of 2:1 and 3:1 give day-to-day thresholds that roughly match the expected day-to-day arrivals, while ratios of 5:1 or higher yield thresholds that begin to mirror the cumulative remaining expected arrivals.

Table 11. Optimal SDSSP1 decision thresholds for a range of blocking costs across three demand arrival scenarios

	Early Arrivals				Middle Arrivals				Late Arrivals			
	Days Before Surgery				Days Before Surgery				Days Before Surgery			
Blocking Costs	4	3	2	1	4	3	2	1	4	3	2	1
1	0	0	0	0	0	0	0	0	0	0	0	0
2	2	1	0	0	1	1	1	1	0	1	1	2
3	3	1	1	1	2	2	2	1	1	2	2	2
4	4	2	1	1	3	2	2	2	2	2	3	3
5	4	2	1	1	3	3	2	2	3	3	3	3
6	4	2	1	1	4	3	3	2	3	3	4	3
7	5	3	2	1	4	4	3	2	4	4	4	4

Interestingly, for lower blocking costs, the thresholds in the late-arriving demand scenario start low and actually increase as the day of surgery approaches. This reflects the fact that deferring a RQ for several days in a row can eventually be as costly as blocking a primary case, suggesting that if the secondary demand is high and arrives early, then a primary specialty with late-arriving demand runs the risk of losing its

space before it has an opportunity to fill it. One way for the OR stakeholders to avoid having a specialty lose its allocated space in this manner is to set a block release date (which is exactly why block release dates are used in practice). However, we note that a blocking cost structure with higher blocking costs before the demand arrival peak and lower costs after the peak could have the same effect by producing appropriate thresholds. It warrants reminding that deferral and blocking costs are not reflective of actual dollar costs, but rather are reflective of subjective satisfaction costs.

The common use of block release dates in practice, and the fact that they often differ between specialties, indicates some underlying, if unstated, complexities in the relative costs that OR managers associate with deferring RQ decisions and blocking primary cases. Based on the results of this analysis of SDSSP1, we argue for an approach that explicitly identifies the relative values placed on blocking and deferring and uses these values to set the corresponding RQ policies. In practice, these values could be made to incorporate a range of practical concerns related to releasing blocks of OR time and scheduling RQ cases. For instance, deferral costs on the days immediately before surgery could reflect potential difficulties in getting last-minute RQ cases cleared for surgery (such as the need for pre-operative tests or payment paperwork). Blocking costs could incorporate equipment requirements and room preferences associated with different primary specialties. Finding reliable methods for eliciting these relative values from the relevant OR stakeholders, and determining how they differ from day-to-day and from specialty-to-specialty, is an area ripe for future research.

We also recall that SDSSP1 reflects a simplified version of the scheduling problem for a single OR where the room capacity, arrival random variables, and RQ decisions are stated in terms of the number of cases, essentially ignoring case durations. The intuitive nature of the threshold result for SDSSP1, and its lack of limitations on the arrival distributions, raises important questions about whether this behavior generalizes to versions of SDSSP with multiple case types (e.g. multiple case durations) and multiple operating rooms. In Chapter 4, we continue our analysis of SDSSP by studying these natural extensions of SDSSP1.

Chapter 4. The Single-Day Surgery Scheduling Problem: Analyzing the General Case

As discussed at the end of the previous chapter, the simple and intuitive nature of the threshold result for SDSSP1 raises the fundamental question of whether this behavior extends to more general versions of SDSSP. In this chapter, we explore two natural extensions of the problem with one OR and unit durations. In Section 4.1 we continue to look at a single OR and study the impact of different case durations. While the threshold behavior is no longer optimal in this case, we show that threshold-based heuristics can be used to generate high quality RQ decisions. In this section we also explore the costs that imposing different block release dates has on scheduling decisions. In Section 4.2, we return to the scenario with unit durations and extend it to consider multiple ORs. We show that the single OR thresholds are optimal when the request queue is sufficiently long and propose a threshold-based heuristic for the case when the RQ is limited in length. Together, these analyses lay a foundation from which to solve realistic problems with multiple ORs and multiple case types.

4.1. A Single OR with Multiple Case Types

In this section we consider the single-day surgery scheduling problem for a single OR with multiple case durations (SDSSP2). The general formulation for SDSSP can be applied directly to SDSSP2 without any modification (simply let $S = 1$). Throughout our discussion of SDSSP2, we will drop the OR index s from our notation (and will continue to vectorize over the case types when convenient). In the following sections we first discuss some implementation issues that are necessary to reduce the computational burden involved in solving the SDP directly. An analysis of optimal policy maps shows that while the solutions to SDSSP2 do not exactly follow a threshold pattern, they do exhibit approximate threshold behavior. This approximate behavior motivates a range of threshold-based heuristics, which we test on a broad range of test problems.

Implementation Issues and Solution Structure

We begin by showing that the day 0 value function for SDSSP2 reduces to an integer knapsack problem. While not critical from the perspective of solving the SDP to optimality (the result integer programs are small and can be solved quite quickly), the knapsack nature of the value function allows us to use knapsack heuristics to find day 0 solutions in our heuristics. Without restating the constraints, the day 0 value function for SDSSP2 can be thus rearranged:

$$
\begin{aligned}
&V_0(C_0, B_0, \boldsymbol{W_0}) \\
&\quad = \min_{\boldsymbol{x_0}} \left\{ \sum_{k=1}^{K} h_0^k \left(\min\left(\left\lfloor \frac{C_0}{d_k} \right\rfloor, W_0^k \right) - x_0^k \right) + r_0 \left(C_0 - \sum_{k=1}^{K} d_k x_0^k \right) \right\} \\
&\quad = r_0 C_0 + \sum_{k=1}^{K} h_0^k \min\left(\left\lfloor \frac{C_0}{d_k} \right\rfloor, W_0^k \right) - \min_{\boldsymbol{x_0}} \left\{ \sum_{k=1}^{K} (h_0^k + d_k r_0) x_0^k \right\} \\
&\quad = r_0 C_0 + \sum_{k=1}^{K} h_0^k \min\left(\left\lfloor \frac{C_0}{d_k} \right\rfloor, W_0^k \right) + \max_{\boldsymbol{x_0}} \left\{ \sum_{k=1}^{K} (h_0^k + d_k r_0) x_0^k \right\}
\end{aligned}
$$

Also recall from the discussion in Section 3.1 that the current formulation places no upper bound on the size of the RQ. In order to implement a solution algorithm for SDSSP2, however, the RQ must be truncated at some point. Early implementations of our code showed that a truncation rule that combined the states with large RQs by ignoring any values beyond the point of truncation created a "bounce-back" effect that impacted the optimal solutions. Apparent irregularities in the optimal solutions followed the location of the truncation. In particular, the solutions for the RQ states closest to the truncation point (no matter where the truncation point was located) would take fewer RQ cases than states with smaller RQs, which ran counter to our results for SDSSP1. This behavior necessitated a more sophisticated technique for limiting the number of RQ states that would not allow the system to prematurely return from the point of truncation.

Instead of truncating, we opt to combine the states with large RQs (those greater than a given boundary) into a state in which the RQ is described as "infinite." In order to justify that this technique does not produce solution irregularities of its own, two

properties must hold. First, once the RQ state variable for a case type exceeds what will fit into the remaining time on the schedule, this condition must continue to hold for all subsequent states. Second, the optimal decisions must be identical for all states beyond the selected boundary. We claim below that these properties hold if the RQ boundary for each case type is located anywhere beyond the range of feasible decisions, which allows the RQ boundaries to be set as small as possible and greatly reduces the computational complexity of the SDP implementation.

Similar to the definitions in Section 3.2, the following definitions will be useful both in the statement and proof of the desired claims. The first definition applies to $j = 1, \dots N$.

$$F_j\left(C_j, B_j, \boldsymbol{W_j}, \boldsymbol{x_j}\right) = \sum_{k=1}^{K} \left(h_j^k ND_j^k + r_j\, E\left[\lambda\, NB_j^k + (1-\lambda) FB_j^k\right]\right) + E\left[V_{j-1}\left(C_{j-1}, B_{j-1}, \boldsymbol{W_{j-1}}\right)\right]$$

$$F_0(C_0, B_0, \boldsymbol{W_0}, \boldsymbol{x_0}) = \sum_{k=1}^{K} h_0^k \left(\min\left(\left\lfloor \frac{C_0}{d_k} \right\rfloor, W_0^k\right) - x_0^k\right) + r_0\left(C_0 - \sum_{k=1}^{K} d_k x_0^k\right)$$

The dependence on the decision variables, x_j^k, is not explicitly stated on the right hand side of the first definition but is implied for all quantities that depend on x_j^k. With these functions thus defined, an alternative statement of the value function for $j = 0, \dots, N$ can be given, with the minimization still subject to the same constraints described above. This re-statement of the value function also provides a concise way to refer to the optimal decision for each feasible state.

$$V_j\left(C_j, B_j, \boldsymbol{W_j}\right) = \min_{\boldsymbol{x_j}}\left\{F_j\left(C_j, B_j, \boldsymbol{W_j}, \boldsymbol{x_j}\right)\right\}$$

$$\boldsymbol{x_j}\left(C_j, B_j, \boldsymbol{W_j}\right) = \arg\min_{\boldsymbol{x_j}}\left\{F_j\left(C_j, B_j, \boldsymbol{W_j}, \boldsymbol{x_j}\right)\right\}$$

Finally, a quantity we refer to as the "trimmed RQ state" is defined for each feasible state that reflects the maximum number of feasible RQ placements for each case type.

$$\widehat{W}_j^k = \min\left(\left\lfloor \frac{C_j}{d_k} \right\rfloor, W_j^k\right)$$

The two lemmas below represent the two properties described above. The first lemma justifies the notion that once the number of RQ cases exceeds the remaining capacity, it will continue to do so. The second lemma states that the value functions and optimal decision vectors are identical for all states where the number of a certain case type on the RQ exceeds what is feasible.

Lemma 4.1 *If* $W_j^k \geq \left\lfloor \frac{C_j}{d_k} \right\rfloor$, *then* $W_{j-1}^k \geq \left\lfloor \frac{C_{j-1}}{d_k} \right\rfloor$ *for all possible future states.*

Proof:

$$W_{j-1}^k - \left\lfloor \frac{C_{j-1}}{d_k} \right\rfloor = \left(W_j^k - x_j^k + R_j^k + (T_j^k - fit_j^k)\right) - \left\lfloor \frac{C_j - \sum_{i=1}^K d_i(x_j^i + fit_j^i)}{d_k} \right\rfloor$$

$$= W_j^k + \left(R_j^k + T_j^k\right) - (x_j^k + fit_j^k) - \left\lfloor \frac{C_j - \sum_{\substack{i=1, \\ i \neq k}}^K d_i(x_j^i + fit_j^i)}{d_k} - \left(x_j^k + fit_j^k\right) \right\rfloor$$

$$= W_j^k + \left(R_j^k + T_j^k\right) - \left\lfloor \frac{C_j - \sum_{\substack{i=1, \\ i \neq k}}^K d_i(x_j^i + fit_j^i)}{d_k} \right\rfloor$$

$$\geq W_j^k - \left\lfloor \frac{C_j}{d_k} \right\rfloor$$

$$\geq 0$$

The first several steps are simply manipulations of the transition equations defined above. The second to last step uses the fact that the arrival random variables are non-negative, as well as an observation on the nature of the floor function. The final step reflects the initial assumption that the desired inequality holds for day j. ■

Lemma 4.2 *For all feasible states* $(C_j, B_j, \boldsymbol{W_j})$ *for all days* $j = 0, \dots, N$*, the following equalities hold:*

(i) $F_j(C_j, B_j, \boldsymbol{W_j}, \boldsymbol{x_j}) = F_j(C_j, B_j, \widehat{\boldsymbol{W}}_{\boldsymbol{j}}, \boldsymbol{x_j})$

(ii) $V_j(C_j, B_j, \boldsymbol{W_j}) = V_j(C_j, B_j, \widehat{\boldsymbol{W}}_{\boldsymbol{j}})$

(iii) $\boldsymbol{x_j}(C_j, B_j, \boldsymbol{W_j}) = \boldsymbol{x_j}(C_j, B_j, \widehat{\boldsymbol{W}}_{\boldsymbol{j}})$

The proof proceeds by induction on j.

Base Case ($j = 0$): The first equality relies directly on the definition of $\widehat{W}_0^k$.

$$\begin{aligned} F_0(C_0, B_0, \boldsymbol{W_0}, \boldsymbol{x_0}) &= \sum_{k=1}^{K} h_0^k \left(\min\left(\left\lfloor \frac{C_0}{d_k} \right\rfloor, W_0^k \right) - x_0^k \right) + r_0 \left(C_0 - \sum_{k=1}^{K} d_k x_0^k \right) \\ &= \sum_{k=1}^{K} h_0^k \left(\widehat{W}_0^k - x_0^k \right) + r_0 \left(C_0 - \sum_{k=1}^{K} d_k x_0^k \right) \\ &= F_0(C_0, B_0, \widehat{\boldsymbol{W}}_{\boldsymbol{0}}, \boldsymbol{x_0}) \end{aligned}$$

Note that the feasible regions for the decision vector are identical for states $(C_0, B_0, \boldsymbol{W_0})$ and $(C_0, B_0, \widehat{\boldsymbol{W}}_{\boldsymbol{0}})$. Both sides of equalities (ii) and (iii) represent minimization problems of identical functions over identical feasible regions, clearly implying that these equalities must hold.

Inductive Step: Assume that all the desired equalities hold for day $j - 1$. Note that moving from state $(C_j, B_j, \boldsymbol{W_j})$ to $(C_j, B_j, \widehat{\boldsymbol{W}}_{\boldsymbol{j}})$ has no impact on the deferral and blocking penalties (ND_j^k, NB_j^k, and FB_j^k), provided that the decision variables are held constant. The deferrals are computed based on what is feasible, the blocking penalties do not rely on the RQ outside of determining feasible decisions, and, as in the base case, the feasible regions are identical for both states. Therefore, looking at the difference between $F_j(C_j, B_j, \boldsymbol{W_j}, \boldsymbol{x_j})$ and $F_j(C_j, B_j, \widehat{\boldsymbol{W}}_{\boldsymbol{j}}, \boldsymbol{x_j})$ comes down to looking at the difference between the subsequent day $j - 1$ states. The next day's available space (C_{j-1}) and blocking eligible hours (B_{j-1}) will also be identical, again provided that the decision variables are held constant. Therefore, the focus lies on potential

differences on day $j-1$ between the RQ states. The goal is to show that the trimmed day $j-1$ RQ states (limited to the day $j-1$ decisions that are feasible) are equal starting from either W_j^k or $\widehat{W}_j^k$ on day j. It will be helpful to express the RQ day j to day $j-1$ transition equation as a function w of the starting RQ state, with all other variables being held constant.

$$w(W_j^k) = \text{day } j-1 \text{ RQ state for case type } k \text{ from state } (C_j, B_j, \boldsymbol{W_j})$$
$$w(\widehat{W}_j^k) = \text{day } j-1 \text{ RQ state for case type } k \text{ from state } (C_j, B_j, \widehat{\boldsymbol{W}}_{\boldsymbol{j}})$$

Showing that these two RQ states have equal trimmed RQ states now amounts to showing that the following equality holds.

$$\min\left(\left\lfloor\frac{C_{j-1}}{d_k}\right\rfloor, w(W_j^k)\right) = \min\left(\left\lfloor\frac{C_{j-1}}{d_k}\right\rfloor, w(\widehat{W}_j^k)\right)$$

If $W_j^k = \widehat{W}_j^k$, then equality trivially holds. If $W_j^k \neq \widehat{W}_j^k$, then $W_j^k > \left\lfloor\frac{C_j}{d_k}\right\rfloor$ and $\widehat{W}_j^k = \left\lfloor\frac{C_j}{d_k}\right\rfloor$. Then by Lemma 4.1, $w(W_j^k) \geq \left\lfloor\frac{C_{j-1}}{d_k}\right\rfloor$ and $(\widehat{W}_j^k) \geq \left\lfloor\frac{C_{j-1}}{d_k}\right\rfloor$. As a result, both sides of the equality reduce to $\left\lfloor\frac{C_{j-1}}{d_k}\right\rfloor$. Returning to equality (i), the difference in question becomes:

$$\begin{aligned} F_j(C_j, B_j, \boldsymbol{W_j}, \boldsymbol{x_j}) - F_j(C_j, B_j, \widehat{\boldsymbol{W}}_{\boldsymbol{j}}, \boldsymbol{x_j}) \\ &= E\left[V_{j-1}\left(C_{j-1}, B_{j-1}, \boldsymbol{w}(\boldsymbol{W_j})\right) - V_{j-1}\left(C_{j-1}, B_{j-1}, \boldsymbol{w}(\widehat{\boldsymbol{W}}_{\boldsymbol{j}})\right)\right] \\ &= E\left[V_{j-1}\left(C_{j-1}, B_{j-1}, \widehat{\boldsymbol{w}(\boldsymbol{W_j})}\right) - V_{j-1}\left(C_{j-1}, B_{j-1}, \widehat{\boldsymbol{w}(\widehat{\boldsymbol{W}}_{\boldsymbol{j}})}\right)\right] \\ &= 0 \end{aligned}$$

The first step uses equality (ii) of the induction assumption, and second uses the result above that the trimmed day $j-1$ RQ states will be equal for both original day j RQ states. As in the base case, equalities (ii) and (iii) follow directly from equality (i). ■

The final two parts of Lemma 4.2 are the justification for combining all states past the RQ boundary into one state. For case type k, $\left\lfloor\frac{C}{d_k}\right\rfloor$ is used as the boundary

because it equals the maximum number of cases of type k that will fit into an empty OR schedule. Because the value function and optimal decisions are identical for all states beyond this boundary (all other things being equal), aggregating these states will not impact the resulting optimal policy.

Optimal Solution Behavior

In order to explore the extent to which the SDSSP1 threshold behavior extends to SDSSP2, we solved SDSSP2 to optimality for a range of input data and analyzed policy maps of the resulting solutions. The policy maps in Figure 5 show the optimal decisions for a single instance of SDSSP2 across a representative selection of states. In this example, the scheduled OR capacity is four hours, and there are two case types: one-hour cases and two-hour cases. The rest of the input data come from one of the test problems described below. The policy maps show the total hours of each case type taken from the RQ and placed on the OR schedule by the optimal decision for each state. A single map shows all possible hours remaining (C_j) and blocking eligible hours (B_j) states for a single RQ state on a particular day. Note that the sum of the hours remaining and the blocking eligible hours cannot exceed the scheduled capacity of the room (as indicated by the infeasible states).

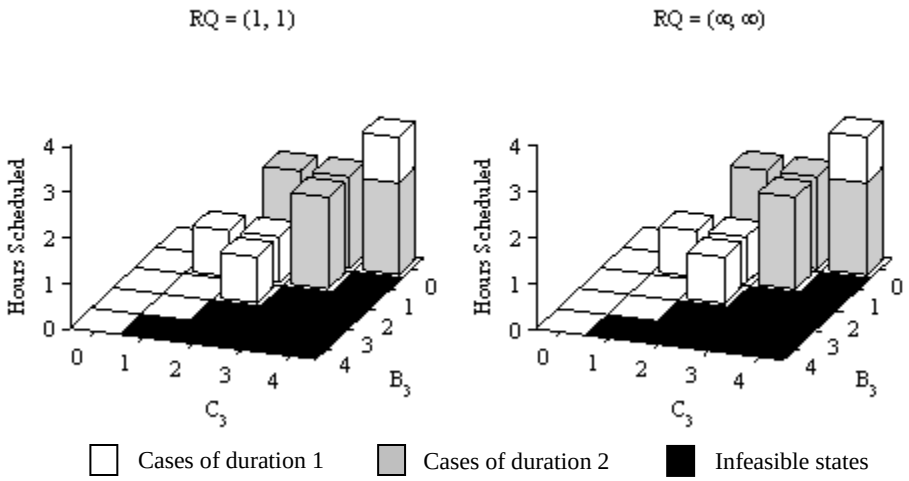

Figure 5. Policy maps for SDSSP2 showing the number of hours of cases scheduled by the optimal decisions three days before surgery for two RQ states across all feasible hours remaining (C_3) and blocking eligible hours (B_3) states.

The fact that the policy map does not change when the number of cases on the RQ is increased confirms the notion (as indicated by Lemma 4.2) that having more cases on the RQ does not pressure the optimal decision into taking more cases. The maps also immediately show that certain behaviors from SDSSP1 do not extend to SDSSP2. In particular, a change in the number of blocking eligible hours can change the optimal decision. This change is evident between $B_3 = 0$ and $B_3 = 1$ for both maps. Figure 6 depicts the number of available hours on the OR schedule after the optimal decisions and clearly illustrates the fact that the decisions do not exactly follow a threshold pattern. However, it does appear that the decisions are guided by a "target" threshold. In this case, the OR manager generally seeks to leave one hour available in the OR, and deviates from this target only by small amounts and in a few situations.

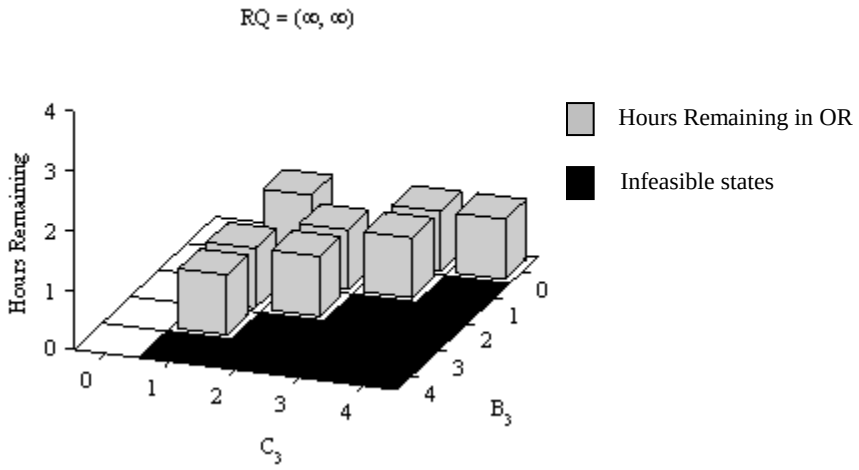

Figure 6. Hours remaining on the OR schedule after the optimal decisions to SDSSP2 three days before surgery for a single RQ state across all feasible hours remaining (C_3) and blocking eligible hours (B_3) states.

The results in Section 3.2 show that the optimal thresholds for SDSSP1 depend on the primary case arrival distribution and on the ratio of blocking costs to deferral costs. In the case of SDSSP2, three other potential factors merit exploration: the ratio of the deferral costs and blocking costs between the case types, the balance of the primary arrivals between the case types, and the weighting parameter (λ) that balances the number blocked (NB_j^k) and the fraction blocked (FB_j^k).

Table 12 shows the ranges of the input data used to create an initial set of 162 test problems. The remaining problem parameters are held constant for all the instances. The scheduled capacity of the OR is set to four hours ($C = 4$), and there are two case types with durations of one and two hours, respectively ($d_1 = 1, d_2 = 2$). The problems are solved over a period of four days before the day of surgery ($N = 4$). The blocking and deferral costs are held constant from day to day for each instance (following the ratios in Table 12), and the cost of unused OR time is set to ten ($r_0 = 10$). The case arrivals on each day follow Poisson distributions, and a total of six hours of primary cases and six hours of secondary cases are expected over the time horizon. For the secondary cases, an equal number of each case type is expected ($E[R_j^1] = E[R_j^2] = 0.5$), and the expected number of primary cases is weighted between the two case types as indicated by the ratios in Table 12. The expected number of primary arrivals is the same every day, implying that this initial set of test problems will not capture the influence of the arrival timing on the optimal solutions. This factor and the impact of block release dates will be explored later.

Table 12. Input data for a set of 162 test problems for the SDSSP2.

Parameter	Notation	Range of Values
Blocking Parameter	λ	0, 0.5, 1
Deferral Cost Ratio	$h_j^2 : h_j^1$	1:1, 2:1
Blocking Cost Ratio	$r_j^2 : r_j^1$	1:1, 2:1, 3:1
Blocking-to-Deferral Cost Ratio	$r_j^1 : h_j^1$	1:1, 3:1, 5:1
Case Type Arrival Rate Ratio	$E[T_j^2] : E[T_j^1]$	1:2, 1:1, 2:1

A thorough analysis of policy maps similar to those in Figure 5 and Figure 6 for each of the test problems reveals insight into three fundamental aspects of the solution behavior: (1) the extent to which the solutions follow an approximate threshold pattern, (2) the nature of the target thresholds (relative to the input data), and (3) the factors that drive the selection of one case type over another. While it is not feasible to present the entire set of policy maps here, we summarize the results of this analysis and present selected policy maps when appropriate.

The most striking and important trend is that the optimal decisions exhibit an approximate threshold behavior like the example shown in Figure 5 and Figure 6. In many cases the thresholds are exact, but even when the optimal solutions deviate from the apparent threshold they do so in only a small number of states and rarely by more than a single hour in either direction.

Of the five factors listed in Table 12, two have no significant impact on the nature of the optimal decisions and apparent target thresholds. As shown by the sample policy maps in Figure 7 for different λ values ($\lambda = 0, 0.5,$ and 1 with all other input held constant), the impact of the blocking weight parameter (λ) is minimal. Smaller λ values effectively lower the blocking costs, and, as a result, more hours of RQ cases tend to be scheduled when $\lambda = 0$ than when $\lambda = 1$. However, the change rarely amounts to more than an hour's difference for a handful of states and never affects the apparent threshold guiding the optimal decisions. Similarly, as shown in Figure 8, the relative prevalence of one primary case type over another ($E[T_j^2]$: $E[T_j^1]$) has very little influence on the optimal decisions. This suggests that the total hours of cases expected each day (which is held constant throughout) is more important in determining the target threshold than the prevalence of a particular case type. Finally, neither factor has any significant impact on the optimal solutions' tendency to prefer one case type over another (i.e. taking two one-hour cases over one two-hour case, or vice versa).

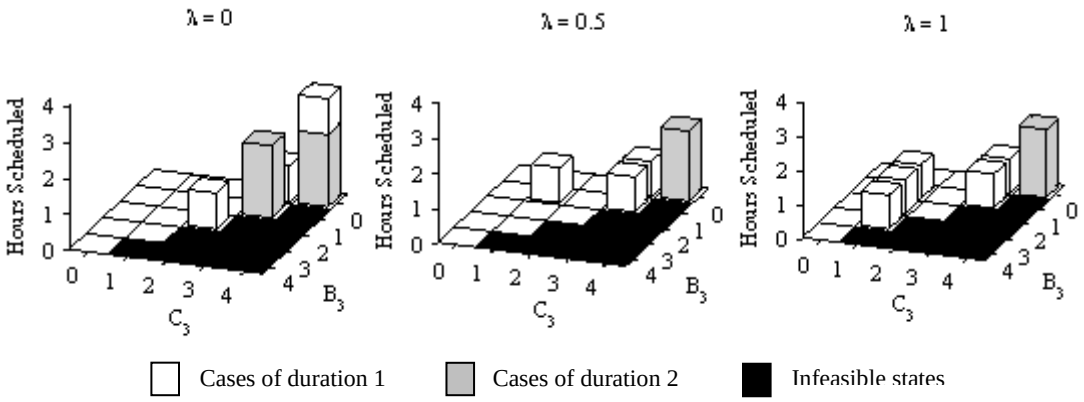

Figure 7. Policy maps for SDSSP2 showing the number of hours of cases scheduled by the optimal decisions three days before surgery for three different blocking weight parameters (λ)

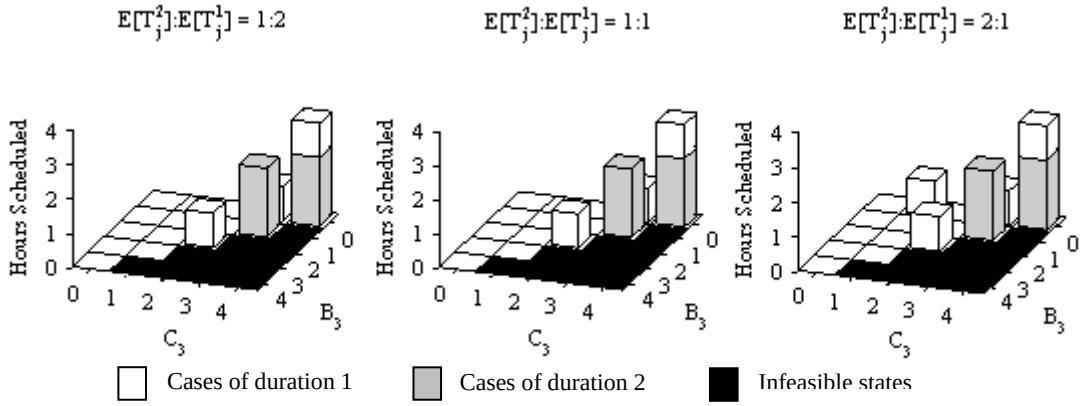

Figure 8. Policy maps for SDSSP2 showing the number of hours of cases scheduled by the optimal decisions three days before surgery for three different primary case type arrival ratios ($E\left[T_j^2\right]: E\left[T_j^1\right]$)

These observations leave the deferral and blocking cost structure as the primary determinant of the target threshold and the selection between the two case types. As in SDSSP1, the ratio of blocking cost to deferral cost (in the form of the $r_j^1: h_j^1$ ratio) has the most influence on the approximate thresholds for the general SDSSP. Furthermore, the apparent thresholds for the three ratios listed in Table 12 (1:1, or *greedy*; 3:1, or *low*; and 5:1, or *high*) exhibit threshold trends much like those observed in Table 11 for SDSSP1. When the two costs are weighted equally, the optimal decisions universally try to fill the OR schedule as much as possible (thus the 'greedy' label). When the ratio is 'low,' the thresholds roughly aim to preserve enough open time for the total hours of cases expected each day. When the ratio is 'high,' the thresholds move to preserve enough open time for the total cumulative remaining hours of expected cases. Sample policy maps for these three scenarios are presented in Figure 9.

The other cost ratios included in this initial set of test problems (the blocking and deferral cost ratios between the case types) also impact the optimal solution behavior. Recall that the cost ratio discussed above is set for the first type ($r_j^1: h_j^1$), but not for the second case type ($r_j^2: h_j^2$). However, the between-case-type ratios ($h_j^2: h_j^1$ and $r_j^2: r_j^1$) affect the ratio of the blocking cost to the deferral cost for the second case type. When the between-case-type ratios combine to the make the cost ratio for the

second case type higher than the cost ratio for the first case type, the approximate thresholds increase by as much as an hour. This pattern is reversed when the between-case-type ratios combine to make the cost ratio lower for the second case type. While these trends in the target thresholds are perhaps not as clean as those observed for SDSSP1, they provide evidence that the optimal solutions to SDSSP2 exhibit threshold behavior similar to that observed for SDSSP1.

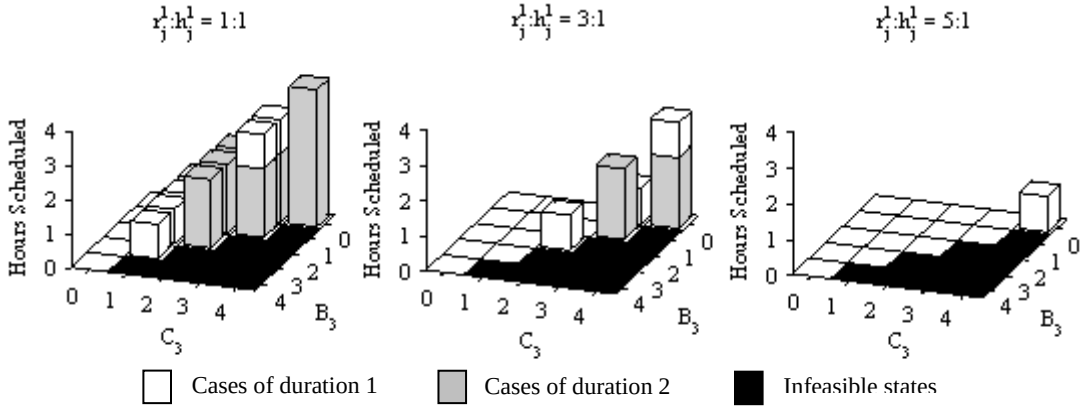

Figure 9. Policy maps for SDSSP2 showing the number of hours of cases scheduled by the optimal decisions three days before surgery for three different blocking to deferral cost ratios ($r_j^1 : h_j^1$)

In some cases the optimal solutions prefer one case type over another. In particular, when the deferral costs are the same for both case types ($h_j^2 : h_j^1 = 1:1$), the solutions uniformly prefer the shorter case type. When the deferral costs are proportional to the case durations ($h_j^2 : h_j^1 = 2:1$), the prioritization becomes more difficult to decipher. However, a weak preference seems to exist for the type with the higher ratio of blocking cost to duration. Together, these observations suggest a prioritization that first considers the ratio of deferral cost to duration and then considers the ratio of blocking cost to duration. Another critical property, particularly in light of the approximate thresholds discussed above, is an overall tendency to aim for the threshold rather than prioritize one case type over another.

These three sets of observations suggest a framework for threshold-based heuristics. As discussed at the end of Section 3.1, the complexity of SDSSP (and thus of SDSSP2) is quadratic in the capacity of the OR and exponential in the number of

case types. This complexity makes heuristic approaches necessary for solving realistically-sized problems (such as the schedule presented in the case study in Section 2.2) in which the scheduled OR capacity may be as large as eight hours and the case durations can exceed four hours.

Threshold-Based Heuristics

Our proposed heuristics for the single-day surgery scheduling problem require two components: (1) a threshold for each day and (2) a case type prioritization rule. The thresholds identify how much open time to preserve for future primary cases and, for each state, define a target number of hours to take off the RQ. The case type prioritization rule then indicates how to reach this target.

The trends in the approximate threshold behavior motivate three simple methods for determining thresholds and a fourth method based on the optimal threshold algorithm from SDSSP1. Each of these methods is described below:

- *Greedy*: Set each threshold to zero hours.
- *Day-to-Day*: Set each threshold to the expected number of hours of cases arriving that day.
- *Cumulative*: Set each threshold to the expected cumulative remaining number of hours of cases.
- *Smart*: Use the optimal threshold algorithm from SDSSP1.

The final method requires some manipulation of the input data in order to apply the SDSSP1 algorithm. Specifically, the multiple case types must be combined into a single case type. The arrival distributions for primary cases are combined to form a single distribution for the hours of arrivals on each day. The deferral and blocking costs for each case type are combined to form weighted costs, with the weights determined by the expected number of arrivals of each case type. The following definitions create the input data for the threshold algorithm. The first and third definitions are for $j = 1, \dots, N$, while the second definition is for $j = 0, \dots, N$.

$$\tilde{T}_j = \sum_{k=1}^{K} d_k T_j^k$$

$$\tilde{h}_j = \frac{\sum_{k=1}^{K} h_j^k E\left[T_j^k\right]}{E\left[\tilde{T}_j\right]}$$

$$\tilde{r}_j = \frac{\sum_{k=1}^{K} r_j^k E\left[T_j^k\right]}{E\left[\tilde{T}_j\right]}$$

Using this modified input, the optimal threshold algorithm from Section 3.2 is used to find the thresholds for the *Smart* threshold rule.

The first three methods, which are based on the observed behavior for specific problem scenarios, should work well for some problems and poorly for others. Specifically, the *Greedy* threshold rule should work well for problems with "greedy" cost ratios. Similarly, the *Day-to-Day* and *Cumulative* threshold rules should work well for problems with the "low" and "high" cost ratios, respectively. The final threshold method can adapt to the input data and we hypothesize that it will perform well in all scenarios.

Three different case type prioritization schemes are based on the insights gleaned from the optimal solution behavior. The last two schemes described below are for days $1, \ldots, N$ only. Because the deferral and blocking cost structure is different on the day of surgery (a single penalty, r_0, for OR hours left empty rather than multiple blocking costs, r_j^k), these ratio dependent prioritizations take advantage of the knapsack nature of the day 0 costs (as described above).

- *Duration*: Case types are prioritized by decreasing duration.
- *Ratios*: Case types are prioritized first by decreasing deferral cost-to-duration ratio, and second by decreasing blocking cost-to-duration ratio. Remaining ties are ordered by decreasing duration.
- *Threshold First*: Cases are prioritized as in the *Ratios* rule, but the prioritization is subjugated to first reaching the target threshold.

This final prioritization scheme is motivated by the observation that the optimal decisions often place a higher priority on reaching the threshold than favoring one

case type over another. However, implementing this final rule requires a simple integer program (IP). In addition to input data on the state of the RQ (W_j^k) and the case types (d_k), the following definitions are needed for the IP:

$D =$ target number of hours to take from the RQ and add to the OR schedule

$p_k =$ priority of case type k

The priority values amount to a permutation of the set of case types $\{1, \dots, K\}$, where a lower priority score denotes a preferred case type. Assuming that the case durations and the scheduled OR capacity are integers, the IP can be defined as follows:

$$\min \quad \sum_{k=1}^{K} (C+1)^{p_k} x_j^k + (C+1)^{K+1} z$$

$$\text{s.t.} \quad \sum_{k=1}^{K} d_k x_j^k + z = D$$

$$0 \le x_j^k \le W_j^k, \quad x_j^k \text{ integer}, \; k = 1, \dots, K$$

$$z \ge 0$$

The slack variable z is given sufficient weight in the objective function to ensure that the target is reached if at all possible. If there is more than one way to hit the target, the RQ variables are weighted according to their priorities to ensure that more preferred cases are selected first. Note that the combination of base and exponent for the objective function weights reflects a kind of base $(C+1)$ arithmetic that ensures that no combination of less-preferred cases will be chosen over a more preferred case type.

As mentioned above, the *Ratio* and *Threshold First* prioritization schemes rely on the knapsack structure of the day 0 costs for the day of surgery. The *Ratio* scheme uses a greedy knapsack heuristic, ordering the case types by decreasing value-to-weight ratio and greedily taking as many of each type as possible. Because the *Threshold First* method already relies on the availability of an IP solver, the day 0 knapsack problem is solved to optimality with this prioritization scheme.

Computational Results

Combining the four threshold determination rules and the three case type prioritization schemes yields twelve threshold-based heuristics. We begin our computational testing with the problem instances defined in Table 12 and explored above. For each problem, we are able to use the optimal solution obtained by solving the SDP and compare it with each of the heuristics. Notice that the blocking weight parameter (λ) and the primary arrivals case type prevalence ($E[T_j^2]:E[T_j^1]$), which have little influence on the optimal solutions, play similarly limited roles in the heuristics. For this reason, the heuristics are tested on the original test instances that have $\lambda = 1$ and $E[T_j^2]:E[T_j^1] = 1:1$. This leaves a set of 18 test problems that vary only in their deferral and blocking cost structure.

An effective way to test the quality of the various heuristics is to imbed them in a simulation environment. The evolution of an OR schedule with specified input data and a RQ decision-making policy can easily be simulated to compute the total cost associated with a RQ policy and a particular realization of the arrival random variables. Each of the heuristics is simulated for 10,000 replications for each of the selected test problems, and the costs associated with each problem are expressed as percentage deviations from the expected value of the optimal solution. Table 13 shows the mean deviation across all replications for each of the heuristics applied to the selected test problems. The 95% confidence interval (CI) half-widths for each of the values shown in Table 13 are given in Table A-1 in the appendix. Note that for all negative entries in the table, zero (representing the expected value of the optimal solution) falls within the CI. Recall that the three simple thresholds (*Greedy*, *Day-to-Day*, and *Cumulative*) are expected to perform better for some instances than others. These targeted instances are shown in italics in Table 13, and the average performance of the heuristics across problem instances is shown for both the targeted instances and for all instances.

Table 13. Mean percentage deviation from the optimal expected cost for twelve threshold-based heuristics applied to test problems across a range of cost structures. Averages across problems are shown both for *targeted* instances and all instances.

$h_j^2 : h_j^1$	1:1										
$r_j^1 : h_j^1$	1:1			3:1			5:1			Mean	
$r_j^2 : r_j^1$	1:1	2:1	3:1	1:1	2:1	3:1	1:1	2:1	3:1	*Target*	All
Greedy											
Duration	*1.3*	*5.2*	*7.9*	19.6	50.3	87.2	61.5	129.2	194.9	*4.8*	61.9
Ratios	*2.8*	*7.7*	*9.4*	19.8	51	88	59.9	128.2	194.8	*6.6*	62.4
Threshold First	*18*	*18.8*	*17.5*	24.6	53.2	88.6	62.4	126.9	191.3	*18.1*	66.8
Day-to-Day											
Duration	50	29.2	15.1	*4.4*	*7.3*	*15.5*	11.4	31.8	48.2	*9.1*	23.7
Ratios	60.1	36.2	21.2	*9.1*	*9.9*	*17.3*	14.7	33.4	47.4	*12.1*	27.7
Threshold First	47.4	27.2	13.2	*2.9*	*5.6*	*14.1*	10	30.6	46.9	*7.5*	22.0
Cumulative											
Duration	96.7	62.2	39	11.6	4.4	3.7	*2.9*	*6.3*	*8.8*	*6.0*	26.2
Ratios	110.6	73.5	47.6	20.1	11	9.2	*10.7*	*12.3*	*16*	*13.0*	34.6
Threshold First	94.6	60.3	37.5	10.4	3.2	2.5	*1.8*	*5.3*	*7.8*	*5.0*	24.8
Smart											
Duration	1.3	31.7	20.8	8.3	4.4	3.7	2.9	4.2	3.8	n/a	9.0
Ratios	2.8	31.5	23	16.2	11	10.5	10.7	11.5	11.8	n/a	14.3
Threshold First	18	36.4	20	7.2	3.2	2.2	1.8	2.6	2.3	n/a	10.4
$h_j^2 : h_j^1$	2:1										
Greedy											
Duration	*-0.2*	*-1.1*	*5.2*	3	24.4	46.8	31.7	75.3	126.6	*1.3*	34.6
Ratios	*7.8*	*-1.1*	*5.2*	5.1	24.4	46.8	33.2	75.3	126.6	*4.0*	35.9
Threshold First	*25.3*	*17.1*	*18.5*	11.3	29.5	50.2	35.4	77.4	127	*20.3*	43.5
Day-to-Day											
Duration	81.7	48.6	35.5	*6*	*2.1*	*4.5*	4.5	13	28.6	*4.2*	24.9
Ratios	89	48.6	35.5	*8.4*	*2.1*	*4.5*	4.8	13	28.6	*5.0*	26.1
Threshold First	81.7	48.6	35.5	*5.9*	*2.1*	*4.5*	4.4	13	28.6	*4.2*	24.9
Cumulative											
Duration	155.1	98.5	74.1	24.6	10.2	4.3	*6.4*	*3.5*	*6.1*	*5.3*	42.5
Ratios	156.8	98.5	74.1	25.2	10.2	4.3	*6.9*	*3.5*	*6.1*	*5.5*	42.8
Threshold First	155.1	98.5	74.1	24.6	10.2	4.3	*6.4*	*3.5*	*6.1*	*5.3*	42.5
Smart											
Duration	-0.2	-1.1	37.3	9.6	9.1	4.3	5.9	3.5	3.2	n/a	8.0
Ratios	7.8	-1.1	37.3	12.1	9.1	4.3	6.4	3.5	3.2	n/a	9.2
Threshold First	25.3	17.1	40	9.7	9.1	4.3	6	3.5	3.2	n/a	13.1

Several observations can be made from these results. First, the scenario-specific thresholds perform well when they should. In most cases, the cost of the solutions is less than 10 percent away from the optimal cost regardless of how the case types are

prioritized. As might be expected, the quality of these thresholds deteriorates rapidly for other cost structures. The *Smart* threshold adapts to the different cost structures, with deviations typically falling at or below 10 percent and only exceeding 20 percent on a few of the "greedy" problems (those with $r_j^1 : h_j^1 = 1:1$). The *Smart* threshold outperforms the other thresholds as a general-purpose heuristic. Furthermore, its flexibility suggests that it will be able to conform and adapt to an even broader range of cost structures than those tested here.

None of the three prioritization schemes significantly outperforms the others across the entire range of problems. The *Threshold First* prioritization scheme is worse than the other prioritization schemes when paired with the *Greedy* and *Smart* thresholds and applied to the "greedy" problems. A greedy approach is appropriate for these problems, as observed from the policy maps, and this discrepancy between the prioritization schemes confirms the intuition that case type preference matters more when larger numbers of RQ cases are being placed. For the other subsets of problems, the *Duration* and *Threshold First* prioritization schemes slightly outperform the *Ratios* rule. Note that for the given case durations (one and two hours) the (decreasing) *Duration* rule will always reach the target if it is possible. This confirms the earlier observation that reaching the threshold is often more important than a particular case type preference.

The thresholds for SDSSP1 suggest that the timing of the primary case arrivals also impacts the target thresholds. In order to study how this affects the performance of the proposed heuristics, a second set of test problems is needed. The full range of inputs used in Table 13 (fixed blocking weight parameter, equal numbers of each case type expected over the course of the problem, and varying cost ratios) is combined with four new arrival patterns. Each of the two case types is set to arrive either "Early" or "Late," and all possible combinations are tested. The arrival random variables are still assumed to follow Poisson distributions for each day, with the "Early" arrival rates set to (1, 0.5, 0.25, 0.25, 0) and the "Late" arrival rates set to (0.25, 0.25, 0.5, 1, 0). This new test set consists of 72 problem instances.

Rather than apply all twelve of the proposed heuristics to this new problem set, three representative heuristics are chosen: *Greedy + Duration*, *Smart + Duration*, and

Smart + Threshold First. The *Greedy + Duration* heuristic is chosen because it is the most naïve of the twelve and serves as a sort of worst case choice. The other two are chosen because they perform well over the entire initial problem set and most closely represent the trends observed in the policy maps. Table 14 shows the average performance of these heuristics over 10,000 simulated replications on the new set of test problems, again shown as the percentage deviation from the optimal solution value. The evenly spread out arrival scenario in Table 13 is referred to as "Middle" and is included in the table for comparison's sake. In the interest of space, the results are only shown for single values of the between-case-type deferral and blocking ratios. Specifically, these ratios in Table 14 are fixed as $h_j^2 : h_j^1 = r_j^2 : r_j^1 = 2:1$ (the case duration ratio). While the exact magnitude of the deviations differs for other deferral and blocking cost structures, the observations and patterns discussed here remain constant across other cost structures. For similar tables for the other cost structures tested, as well the 95% CI half-widths for all the means, please refer to Table A-7 through Table A-12 in the appendix.

Many of the insights from the initial set of test problems are still evident in this second set of tests. As expected, the *Smart* thresholds tend to outperform the *Greedy + Duration* heuristic in all arrival scenarios except the "greedy" problems where the *Greedy* threshold is expected to perform well. On average, the *Smart + Duration* combination still slightly outperforms the *Smart + Threshold First* combination. Most of difference between these two heuristics originates from the "greedy" problems, while their performance is nearly identical for the other cost ratios.

It is also clear from the results in Table 14 that the timing of the primary service line's arrivals does have some impact on the performance of the heuristics. The *Greedy + Duration* heuristic performs better overall when more of the cases arrive early, while the other two heuristics perform worst in the (Early, Early) scenario. The early arrivals leave fewer arrivals closer to the day of surgery, which reduces the time that must be preserved for remaining cases and makes greedier policies more appropriate. As observed with the first set of test problems, the *Smart* threshold suffers in scenarios where greedier policies are appropriate and more importance is placed on the case type preference. On the other hand, the *Smart* threshold paired

with either case type preference does well when lower thresholds are required. In these scenarios, the case type preference plays a less significant role because fewer hours of cases are taken. While these observations appear at first to implicate the case type preferences as the weak step of the proposed heuristics, it may in fact be the case that the threshold behavior itself becomes less precise as the optimal solutions become greedier.

From the perspective of the SDSSP, block release dates act as constraints on RQ decisions and should lead to lower quality solutions. However, the extent of the impact will depend upon the nature of the optimal decisions. In particular, forcing some decisions to zero will have less impact on problems with high target thresholds and will have more impact on problems with low target thresholds. In order to study the impact of block release dates, the 90 test problems presented in Table 13 and Table 14 are combined to form a single set of problem instances. The feasible block release dates for these problems range from three days before surgery (day 3) to the day of surgery (day 0). Setting the block release date to day 3 does not constrain the solutions in any way because day 3 is the first day on which a RQ decision can be made (the system is empty on day 4). Once a block release date has been chosen, a RQ policy is needed to make decisions after the block is released. Three different RQ policies are combined with the block release dates: (1) the optimal decisions from the fully solved SDPs, (2) the *Greedy + Duration* heuristic, and (3) the *Smart + Duration* heuristic. Each of these twelve combinations is simulated over 10,000 replications for each problem instance and the average results are shown in Table 15. The results are again given as percentage deviations from the unconstrained optimal solutions. As in Table 14, the results are only shown for single values of the between-case-type deferral and blocking ratios ($h_j^2 : h_j^1 = r_j^2 : r_j^1 = 2:1$). The exact magnitude of the deviations for other deferral and blocking cost structures differs some, but the observations and patterns discussed here remain constant across the other cost structures. For tables showing the results of our tests for these other cost structures, as well as the 95% confidence interval half-widths for all the means, please see Table A-18 through Table A-23 in the appendix.

Table 14. Mean percentage deviation from the optimal solution for three threshold-based heuristics applied to test problems representing a range of primary service line arrival patterns and cost structures (for $h_j^2 : h_j^1 = r_j^2 : r_j^1 = 2:1$)

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Greedy + Duration	1.9	15.8	40	-1.1	24.4	75.3	2.3	21.4	77.0	1.8	17.6	60.1	0.4	15.0	61.5
Smart + Duration	1.9	20.1	12	-1.1	9.1	3.5	2.3	6.4	0.4	1.8	5.5	3.5	0.4	1.3	3.8
Smart + Threshold First	14.7	20.3	12.1	17.1	9.1	3.5	28.8	6.4	0.4	18.3	5.5	3.5	19.3	1.3	3.8

Table 15. Mean percentage deviation from the unconstrained optimal solution for three RQ policies paired with different block release dates across a range of primary service line arrival patterns and cost structures (for $h_j^2 : h_j^1 = r_j^2 : r_j^1 = 2:1$)

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Optimal Decisions															
Day 3	2.9	-0.3	-1.6	-1.8	1.4	2.4	-0.9	0.5	0.6	2.6	1.4	-1.0	-1.1	-2.1	-0.6
Day 2	42.1	5.4	-1.3	38.0	6.1	2.5	47.5	8.5	0.8	44.6	7.0	-0.6	44.3	4.3	-0.3
Day 1	93.1	19.9	3.3	81.2	9.8	2.6	100.4	10.9	0.8	92.5	12.6	-0.5	100.6	11.7	1.3
Day 0	143.0	35.3	7.1	115.0	12.8	4.2	124.9	11.1	0.8	123.3	16.7	0.2	141.5	17.3	3.9
Greedy + Duration															
Day 3	5.3	15.2	39.8	0.7	23.6	75.2	1.6	22.5	77.0	8.2	19.9	58.2	-0.5	14.9	59.8
Day 2	43.4	8.2	8.9	40.6	21.4	52.0	50.8	37.8	83.3	52.5	27.9	52.1	44.8	15.1	41.4
Day 1	94.5	21.4	8.1	83.2	16.0	25.6	103.6	36.0	54.7	97.8	28.3	31.5	101.5	23.4	31.3
Day 0	143.0	35.3	7.1	115.0	12.8	4.2	124.9	11.1	0.8	123.3	16.7	0.2	141.5	17.3	3.9
Smart + Duration															
Day 3	5.3	17.1	9.2	0.7	8.2	5.5	1.6	8.8	0.8	8.2	8.1	2.7	-0.5	-0.6	1.5
Day 2	43.4	25.7	8.9	40.6	9.3	5.5	50.8	11.1	0.8	52.5	18.4	2.7	44.8	5.9	1.5
Day 1	94.5	34.9	12.6	83.2	9.8	3.5	103.6	11.1	0.8	97.8	18.4	2.7	101.5	13.8	2.2
Day 0	143.0	35.3	7.1	115.0	12.8	4.2	124.9	11.1	0.8	123.3	16.7	0.2	141.5	17.3	3.9

When paired with the optimal decisions, the day 3 results are consistently within 3 percent of the optimal expected value (with the 95% CI half-widths always including zero), verifying the simulation's ability to accurately gauge the performance of a given policy. When paired with the heuristic RQ policies, the day 3 results are directly in line with the unconstrained results presented in Table 13 and Table 14, confirming for these instances that setting the block release date to day 3 is equivalent to having no block release date.

As the block release date moves closer to the day of surgery, its impact on solution quality becomes evident. Note that when the block release date is set to day 0, the choice of RQ policy loses significance because all the policies become greedy on the day of surgery. For the problems where greedier solutions are appropriate ($r_j^1 : h_j^1 = 1:1$), the solutions deteriorate quickly as the block release policy becomes more restrictive. For other blocking-to-deferral cost ratios, the added costs of the block release dates depend heavily on the RQ policy they are paired with. When the block release dates are paired with good RQ policies (the optimal decisions or *Smart + Duration*), the added costs increase slightly as the block release policy becomes more restrictive, but rarely exceed 20 percent deviation from the optimal. In contrast, when the block release dates are paired with a poor RQ policy (*Greedy + Duration*) the solutions at times improve as the block release policy becomes more restrictive. This is most noticeable when the blocking-to-deferral cost ratio is high ($r_j^1 : h_j^1 = 5:1$), which is exactly the scenario where the *Greedy* threshold performs poorly. This ability of block release dates to mitigate the effect of poor RQ policies adds a level of theoretical justification to their use in practice, where the overall quality of RQ policies may be unknown and difficult to ascertain.

Discussion

The results of this section demonstrate that the threshold behavior that was shown to be optimal for SDSSP1 can be successfully extended to yield high quality solutions to the surgery scheduling problem with multiple case durations (SDSSP2). Showing that the optimal solutions to SDSSP2 remain constant for large numbers of cases on the RQ maintains the computational tractability of the SDP and allows a diverse

range of test problems to be solved to optimality. A detailed analysis of the optimal policy maps for these test problems reveals that the solutions follow an approximate threshold behavior and gives insight into how good thresholds and case type prioritization rules can be combined to form threshold-based heuristics.

A set of threshold-based heuristics is proposed, and the computational results show that heuristics using the optimal threshold algorithm from SDSSP1 consistently outperform heuristics using other threshold rules. While none of the proposed case type prioritization schemes dominates the others, the results favor schemes that focus on hitting the target thresholds rather than enforcing strict case type preferences. The nature of the target thresholds varies with both the deferral and blocking cost structure and the timing of the primary case arrivals. The results indicate that the threshold-based heuristics perform best when the target thresholds are higher (indicating that fewer RQ cases will be scheduled) and deteriorate somewhat when greedy (lower) target thresholds are appropriate.

Block release dates are shown to be increasingly costly for problems with greedy target thresholds. However, in scenarios with higher target thresholds, imposing block release dates is not overly costly if good RQ decisions are made after the block is released. Block release dates can improve solution quality when paired with poor RQ decisions (that is, the resulting solutions are not as bad as they would be otherwise). While the results of this section have focused on extending SDSSP1 to problems with multiple case durations, it is equally important to explore how SDSSP1 can be generalized to problems with multiple operating rooms. In the next section, we consider this second extension.

4.2. Multiple ORs with Unit Durations

The previous section showed that the optimal thresholds from the problem with a single OR and unit durations (SDSSP1) can still be used to make good, if not optimal, RQ decisions for the problem with a single OR and multiple case durations. The other natural extension to SDSSP1 is to look at problems with multiple ORs while maintaining the assumption that all cases have unit durations. We refer to this extension as SDSSP3 and investigate in this section the extent to which the optimality

of the single OR thresholds (as studied in Section 3.2) can be extended to the problem with multiple rooms.

If we return to the general formulation in Section 3.1, we see that the only factor linking the different operating rooms together is the presence of a single, shared RQ. A primary role of this shared RQ state in the SDP value function is to limit the total number of RQ cases that can be added to the OR schedules. For this reason, we explore SDSSP3 under two scenarios. In the first scenario, we assume that the RQ is (essentially) infinite, and show that the under this assumption SDSSP3 can separated into distinct problems for each OR and solved using the respective optimal SDSSP1 thresholds. In the second scenario, we consider the case where the RQ is limited. In this scenario, we show that SDSSP3 cannot be separated and, furthermore, that the structure that facilitated the proof of Theorem 3.1 does not quite hold in the case of multiple ORs. However, these attempts suggest an intuitive heuristic approach that uses the SDSSP1 data for each OR to find solutions for SDSSP3.

Before continuing, we observe that the formulation for SDSSP3 can be obtained from the general SDSSP formulation by setting $K = 1$ and $d_1 = 1$. In the analysis that follows, we drop the case type index k from our notation for clarity's sake, and vectorize over the different ORs when appropriate. Also note that the lack of different case durations eliminates the need for the *ProcessDay* algorithm and allows us to use the simplified expressions for NB_j, C_{j-1}, B_{j-1}, and W_{j-1} (appropriately indexed to account for different ORs) from Section 3.2. As with our earlier analyses, it is helpful to define the individual functions that the SDP value and boundary functions are trying to minimize. The first definition applies for $j = 1, \dots, N$, and the second applies for $j = 0$.

$$F_j(\boldsymbol{C_j}, \boldsymbol{B_j}, W_j, \boldsymbol{x_j}) = h_j ND_j + \sum_{s=1}^{S} r_j^s E\left[NB_j^s\right] + E\left[V_{j-1}(\boldsymbol{C_{j-1}}, \boldsymbol{B_{j-1}}, W_{j-1})\right]$$

$$F_0(\boldsymbol{C_0}, \boldsymbol{B_0}, W_0, \boldsymbol{x_0}) = h_0 \left[\min\left(W_0, \sum_{s=1}^{S} C_0^s\right) - \sum_{s=1}^{S} x_0^s\right] + \sum_{s=1}^{S} r_0^s (C_0^s - x_0^s)$$

Infinite Request Queue

Our initial analysis of SDSSP3 will consider the scenario where the RQ is infinite. As with SDSSP2, the concept of an "infinite" RQ can be interpreted as a RQ that is large enough to be beyond the feasibility limits imposed by the capacity of the ORs. While we will not present this concept formally for SDSSP3, a similar approach to the one used in Section 4.2 can be used. In the analysis that follows, we will use the notation $W_j = \infty$ to represent this assumption.

In order to show that SDSSP3 can be solved to optimality by breaking it into separate instances of SDSSP1 for each OR, it is sufficient to show that the function $F_j(\boldsymbol{C_j}, \boldsymbol{B_j}, \infty, \boldsymbol{x_j})$ and the value function $V_j(\boldsymbol{C_j}, \boldsymbol{B_j}, \infty)$ can be separated into distinct functions for each OR.

Lemma 4.3 *If $W_j = \infty$ for $j = 0, \dots N$, then the function $F_j(\boldsymbol{C_j}, \boldsymbol{B_j}, \infty, \boldsymbol{x_j})$ and the value function $V_j(\boldsymbol{C_j}, \boldsymbol{B_j}, \infty)$ for SDSSP3 are separable by operating room such that:*

$$(i)\ F_j(\boldsymbol{C_j}, \boldsymbol{B_j}, \infty, \boldsymbol{x_j}) = \sum_{s=1}^{S} F_j^s(C_j^s, B_j^s, \infty, x_j^s) \quad \textit{and}$$

$$(ii)\ V_j(\boldsymbol{C_j}, \boldsymbol{B_j}, \infty) = \sum_{s=1}^{S} V_j^s(C_j^s, B_j^s, \infty)$$

The proof proceeds by induction on j.

Base Case ($j = 0$):

$$\begin{aligned} F_0(\boldsymbol{C_0}, \boldsymbol{B_0}, \infty, \boldsymbol{x_0}) &= h_0 \left[\sum_{s=1}^{S} C_0^s - \sum_{s=1}^{S} x_0^s \right] + \sum_{s=1}^{S} r_0^s (C_0^s - x_0^s) \\ &= \sum_{s=1}^{S} (h_0 + r_0^s)(C_0^s - x_0^s) \\ &=: \sum_{s=1}^{S} F_0^s(C_0^s, B_0^s, \infty, x_0^s) \end{aligned}$$

Observe that the function $F_0^s(C_0^s, B_0^s, \infty, x_0^s)$ is defined exclusively in terms of room s and exactly matches the corresponding function from SDSSP1 with $W_0 = \infty$. With $W_0 = \infty$, the only constraints on the day 0 minimization of $F_0(\boldsymbol{C_0}, \boldsymbol{B_0}, \infty, \boldsymbol{x_0})$ are $0 \le x_0^s \le C_0^s$ for each room s. Thus the function $F_0^s(C_0^s, B_0^s, \infty, x_0^s)$ can be minimized for each room without respect to the decisions made in other rooms.

$$
\begin{aligned}
V_0(\boldsymbol{C_0}, \boldsymbol{B_0}, \infty) &= \min\{F_0(\boldsymbol{C_0}, \boldsymbol{B_0}, \infty, \boldsymbol{x_0}) \mid 0 \le x_0^s \le C_0^s \quad \forall s\} \\
&= \min\left\{\sum_{s=1}^{S} F_0^s(C_0^s, B_0^s, \infty, x_0^s) \mid 0 \le x_0^s \le C_0^s \quad \forall s\right\} \\
&= \sum_{s=1}^{S} \min\{F_0^s(C_0^s, B_0^s, \infty, x_0^s) \mid 0 \le x_0^s \le C_0^s\} \\
&=: \sum_{s=1}^{S} V_0^s(C_0^s, B_0^s, \infty)
\end{aligned}
$$

Inductive Step: Assume that value function is separable by room for each feasible state with $W_j = \infty$ on day $j-1$. Because $W_j = \infty$, the number deferred on day j (ND_j) can be separated into the number deferred as a result of each room's decision, just as the blocking penalties incurred by a particular OR have no impact on the blocking penalties incurred by the other ORs.

$$
\begin{aligned}
F_j\left(\boldsymbol{C_j}, \boldsymbol{B_j}, \infty, \boldsymbol{x_j}\right) &= h_j ND_j + \sum_{s=1}^{S} r_j^s E\left[NB_j^s\right] + E\left[V_{j-1}\left(\boldsymbol{C_{j-1}}, \boldsymbol{B_{j-1}}, \infty\right)\right] \\
&= h_j \sum_{s=1}^{S} \left(C_j^s - x_j^s\right) + \sum_{s=1}^{S} r_j^s E\left[NB_j^s\right] + E\left[\sum_{s=1}^{S} V_{j-1}^s\left(C_{j-1}^s, B_{j-1}^s, \infty\right)\right] \\
&= \sum_{s=1}^{S} \left(h_j\left(C_j^s - x_j^s\right) + r_j^s E\left[NB_j^s\right] + E\left[V_{j-1}^s\left(C_{j-1}^s, B_{j-1}^s, \infty\right)\right]\right) \\
&=: \sum_{s=1}^{S} F_j^s\left(C_j^s, B_j^s, \infty, x_j^s\right)
\end{aligned}
$$

Again, the room specific functions $F_j^s\left(C_j^s, B_j^s, \infty, x_j^s\right)$ are defined exactly as in SDSSP1 with $W_j = \infty$. Repeating the argument made in the base case, the feasible

region for the decision variables can be separated by room, and the desired separability of the value function for day j follows.

$$V_j(\boldsymbol{C_j}, \boldsymbol{B_j}, \infty) = \sum_{s=1}^{S} \min\{F_j^s(C_j^s, B_j^s, \infty, x_j^s) \mid 0 \le x_j^s \le C_j^s\}$$
$$=: \sum_{s=1}^{S} V_j^s(C_j^s, B_j^s, \infty)$$

This completes the proof of the lemma. ■

The ability to separate the value function by room implies that SDSSP3 with an infinite RQ can be solved by solving SDSSP1 for each room individually. If the optimal thresholds for SDSSP1 for room s are defined as $\{Y_j^s \text{ for } j = 0, \dots, N\}$, then the optimal SDSSP3 decision for room s on day j is given by:

$$x_j^s(\boldsymbol{C_j}, \boldsymbol{B_j}, \infty) = \max(0, C_j^s - Y_j^s)$$

Limited Request Queue

The result above clearly indicates that the optimal SDSSP1 thresholds for each operating room have a role to play in the optimal solutions to SDSSP3. Unfortunately, the separability argument used in Lemma 4.3 breaks down for finite RQ states for two reasons. First, the feasible region for the decision vector can no longer be separated by room. Second, even though the day j costs associated with a given decision can be separated by room, the subsequent day $j-1$ value function cannot be separated (because of its continued dependence on the shared RQ state W_{j-1}). Furthermore, if we naïvely try to use the SDSSP1 thresholds in the case where the RQ is not infinite, we clearly run into difficulties when the number of cases on the RQ is not sufficient to meet the sum of the RQ decisions dictated by the individual thresholds (that is, when $W_j < \sum_{s=1}^{S} \max(0, C_j^s - Y_j^s)$). The uncertainty involved in moving the unfeasible threshold-based decision back to the feasible region is illustrated for two ORs in Figure 10. How should the limited number of RQ cases be distributed between the rooms? Should one threshold be met and other not met, or

should the cases be evenly distributed between the rooms? More importantly, if we ignore the constraints on day j's value function minimization, is there any mathematical justification for believing that the SDSSP1 thresholds continue to give the optimal decisions to the unconstrained S-dimensional minimization problem?

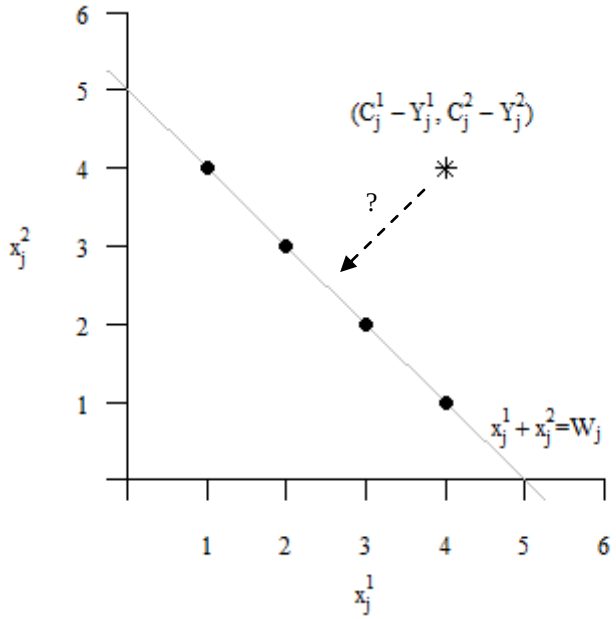

Figure 10. Illustration of the limitations the shared request queue places on meeting the individual operating room thresholds for SDSSP3.

Recall from the proof of Theorem 3.1 that the key to determining each day's optimal threshold is the structure we found in the finite difference functions (in particular, the theorem claims that $\Delta F_j(C_j, B_j, W_j, x_j) = G_j(C_j - x_j)$ for some function $G_j(n)$ that is identical for all states on day j). A logical step, then, for the S-dimensional minimization problem associated with SDSSP3 is to look at the partial finite differences associated with each room. Using the notation $\boldsymbol{e_i}$ to refer to the unit vector in the ith dimension, we can define these partial finite differences as follows:

$$\Delta_s F_j(\boldsymbol{C_j}, \boldsymbol{B_j}, W_j, \boldsymbol{x_j}) = F_j(\boldsymbol{C_j}, \boldsymbol{B_j}, W_j, \boldsymbol{x_j} + \boldsymbol{e_s}) - F_j(\boldsymbol{C_j}, \boldsymbol{B_j}, W_j, \boldsymbol{x_j})$$

Ideally, we would like to be able to show that these partial finite differences exhibit the same structure as the finite differences in the proof of Theorem 3.1. If this were the case, we would know that the unconstrained minimization of $F_j(\boldsymbol{C_j}, \boldsymbol{B_j}, W_j, \boldsymbol{x_j})$

(which would then be a convex function) could still use the SDSSP1 thresholds for each OR (because the partial for each room would be independent of the other rooms). Furthermore, the partial finite difference functions could then be used in a form of "slowest ascent" algorithm (in the spirit of the classical steepest descent optimization algorithm) to move from the unconstrained minimum to the constrained minimum.

As we continue with the calculation of these partial finite differences, it is important to note that the group 1, 2, and 3 transition relationships for NB_j, C_{j-1}, B_{j-1}, and W_{j-1} continue to hold in the multiple OR case. Also, changing the decision value for one of the ORs does not change the blocking costs incurred by the other ORs. The following calculations show that the partial finite differences do not exhibit the same structure as the finite differences for each individual room.

$$\begin{aligned}
\Delta_s F_j(\boldsymbol{C_j}, \boldsymbol{B_j}, W_j, \boldsymbol{x_j}) &= F_j(\boldsymbol{C_j}, \boldsymbol{B_j}, W_j, \boldsymbol{x_j} + \boldsymbol{e_s}) - F_j(\boldsymbol{C_j}, \boldsymbol{B_j}, W_j, \boldsymbol{x_j}) \\
&= h_j\left[ND_j(\boldsymbol{x_j} + \boldsymbol{e_s}) - ND_j(\boldsymbol{x_j})\right] + r_j^s\left[NB_j^s(\boldsymbol{x_j} + \boldsymbol{e_s}) - NB_j^s(\boldsymbol{x_j})\right] \\
&\quad + E\Big[V_{j-1}\left(\boldsymbol{C_{j-1}}(\boldsymbol{x_j} + \boldsymbol{e_s}), \boldsymbol{B_{j-1}}(\boldsymbol{x_j} + \boldsymbol{e_s}), W_{j-1}(\boldsymbol{x_j} + \boldsymbol{e_s})\right) \\
&\qquad - V_{j-1}\left(\boldsymbol{C_{j-1}}(\boldsymbol{x_j}), \boldsymbol{B_{j-1}}(\boldsymbol{x_j}), W_{j-1}(\boldsymbol{x_j})\right)\Big] \\
&= -h_j + r_j^s P\left[T_j^s \geq C_j^s - x_j^s\right] \\
&\quad + E\Big[V_{j-1}\left(\boldsymbol{C_{j-1}}(\boldsymbol{x_j} + \boldsymbol{e_s}), \boldsymbol{B_{j-1}}(\boldsymbol{x_j} + \boldsymbol{e_s}), W_{j-1}(\boldsymbol{x_j} + \boldsymbol{e_s})\right) \\
&\qquad - V_{j-1}\left(\boldsymbol{C_{j-1}}(\boldsymbol{x_j}), \boldsymbol{B_{j-1}}(\boldsymbol{x_j}), W_{j-1}(\boldsymbol{x_j})\right)\Big]
\end{aligned}$$

So far, this calculation exactly follows the form of the $G_j(n)$ calculation in the inductive step of the proof of Theorem 3.1 for SDSSP1. In fact, the similarities continue with the difference between the day $j-1$ value functions. The group 1 transition relationships imply that when $T_j^s \geq C_j^s - x_j^s$, the day $j-1$ states $\left(\boldsymbol{C_{j-1}}(\boldsymbol{x_j} + \boldsymbol{e_s}), \boldsymbol{B_{j-1}}(\boldsymbol{x_j} + \boldsymbol{e_s}), W_{j-1}(\boldsymbol{x_j} + \boldsymbol{e_s})\right)$ and $\left(\boldsymbol{C_{j-1}}(\boldsymbol{x_j}), \boldsymbol{B_{j-1}}(\boldsymbol{x_j}), W_{j-1}(\boldsymbol{x_j})\right)$ are identical. Therefore, the difference between the day $j-1$ value functions is only nonzero when $T_j^s < C_j^s - x_j^s$. However, a relationship similar to the one in part (iv) of Theorem 3.1 (expressing the difference day $j-1$ value functions in terms of room

s's partial finite difference) cannot be found because of the continued interdependence of the ORs and the RQ on day $j-1$.

Even though this approach falls short theoretically, the strong similarities between the calculation of $\Delta_s F_j(\boldsymbol{C_j}, \boldsymbol{B_j}, W_j, \boldsymbol{x_j})$ and the finite difference functions $G_j(n)$ for each individual room suggest that the single OR finite differences can still be used as reasonable approximations to the multiple OR partial finite differences. If we continue to define Y_j^s and $G_j^s(n)$ for $j = 0, \ldots, N$ as the single OR thresholds and finite difference functions emerging from Theorem 3.1 applied to room s, then we can approximate $\Delta_s F_j(\boldsymbol{C_j}, \boldsymbol{B_j}, W_j, \boldsymbol{x_j})$ as follows:

$$\Delta_s F_j(\boldsymbol{C_j}, \boldsymbol{B_j}, W_j, \boldsymbol{x_j}) \approx G_j^s(C_j^s - x_j^s)$$

$$= -h_j + r_j^s P[T_j^s \geq C_j^s - x_j^s] + \sum_{i=1}^{Y_{j-1}^s} P[T_j^s = C_j^s - x_j^s - i] G_{j-1}^s(i)$$

Using these approximations, the optimal SDSSP1 threshold data can be still be used in the "slowest ascent" heuristic approach proposed above. While not guaranteed to be optimal, we now at least have some justification that this approach behaves in a way that reasonably approximates the behavior of the actual partial finite differences.

The Threshold-Based Slowest Ascent Heuristic

In this subsection we propose a slowest ascent heuristic that uses the data from the optimal SDSSP1 threshold algorithm for each individual OR to generate approximate solutions for SDSSP3. The spirit of the algorithm is presented in Figure 10 and Figure 11. Suppose we are in the situation shown in Figure 10 for two ORs, with the optimal SDSSP1 thresholds indicating a preference for taking more cases off the RQ than are feasible. The added cost of reducing the decision by one case in either direction can be found using the partial finite difference for the respective direction. Using the proposed approximation for the partials, our heuristic reduces the threshold-based decision by one case in the direction that has the smallest marginal cost (i.e. the direction that gives the "slowest ascent"). If the reduced decision is still not feasible, then repeated slowest ascent steps can be taken until the decision reaches the feasible

region. This iterative process is more clearly defined in the pseudocode for the proposed heuristic given below. We note that in applying the heuristic to a given state $(\boldsymbol{C_j}, \boldsymbol{B_j}, W_j)$ on a given day j, we assume that the optimal SDSSP1 thresholds $\{Y_j^s, \text{for } s = 1, \dots, S\}$ and the finite difference function evaluations $\{G_j^s(n), \text{for } n = 0, \dots, C, s = 1, \dots, S\}$ have been calculated and are available.

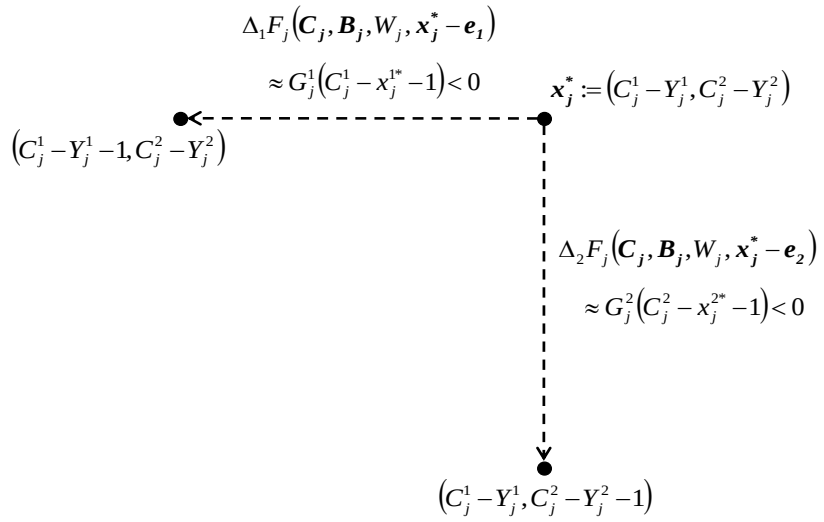

Figure 11. Example of the data used during the slowest ascent decision process for SDSSP3 with two operating rooms

Pseudocode for the *Threshold-Based Slowest Ascent Heuristic*

INPUT:

$(\boldsymbol{C_j}, \boldsymbol{B_j}, W_j)$, $\{Y_j^s, \text{for } s = 1, \dots, S\}$, and $\{G_j^s(n), \text{for } n = 0, \dots, C, s = 1, \dots, S\}$

INITIALIZE:

Set $x_j^{s*} = \max(0, C_j^s - Y_j^s)$ for $s = 1, \dots, S$ // Unconstrained by the RQ

Define $A = \{i \mid x_j^i > 0\}$ // Rooms eligible for reduction

WHILE $\sum_{s=1}^{S} x_j^s > W_j$:

// Step in direction of room with lowest marginal increase in cost

Find the room $i \epsilon A$ with the minimal value of $\left| G_j^i \left(C_j^i - \left(x_j^{i*} - 1 \right) \right) \right|$

Set $x_j^{i*} = x_j^{i*} - 1$ and if $x_j^{i*} = 0$ remove i from set A

END-WHILE

RETURN: x_j^{s*} for $s = 1, \dots, S$

While we did not test this algorithm computationally, the calculations above together with Lemma 4.3 give us reason to hypothesize that this threshold-based heuristic will find high quality decisions for SDSSP3.

4.3. Surgery Scheduling Conclusions

Our study of the single-day surgery scheduling problem is the first to model the sequential nature of an OR manager's daily RQ decisions and to investigate the dynamic interaction of these decisions with the block schedule, block release dates, and primary specialty arrival patterns. We first obtain theoretical, threshold-based results for a special case of the surgery scheduling problem, and then systematically analyze how these results can be applied in more general settings. In particular, the analysis in Section 3.2 shows how an OR manager can make equitable RQ decisions by setting daily threshold targets that define how much time to preserve for future primary demand. Section 4.1 studies how this concept applies to problems with multiple case durations, while Section 4.2 proposes a heuristic that helps to prioritize between different ORs' thresholds when the RQ does not have enough cases to meet them all individually. Finally, our computational testing in Section 4.1 shows that block release dates, which generally work as constraints on OR managers' decisions, can serve as practical safeguards against the costs associated with poor RQ decisions (i.e. decisions that do not follow threshold-based guidelines).

A common theme throughout our analysis, from the case study in Section 2.2 through the analyses in this chapter, is the importance of both the primary demand arrival patterns and the relative blocking and deferral costs associated with request queue and block release decisions. The type of data collected in our observation of UMMC's scheduling system illustrates the kind of empirical analysis that can be done with surgical suite data to estimate the demand that a particular specialty or surgeon generates for an allocated block of OR time. As mentioned at the end of Chapter 3, the relative costs that OR stakeholders associate with blocking and deferral penalties have the potential to be quite complex, varying from day-to-day and from specialty-to-specialty and reflecting a wide range of practical concerns. Our research has shown how sensitive the proposed threshold-based rules are to these relative costs, and

developing reliable methods for eliciting these costs from the appropriate personnel is an important area for future research. While these final comments point out the need for continued research before the proposed threshold-based rules can be implemented in realistic OR scheduling settings, we feel that the research presented in these chapters succeeds in finding significant insights into a previously understudied aspect of surgery scheduling.

Chapter 5. Mass Casualty Terrorist Bombing Response Planning: Literature Review

The management of the medical response to mass casualty incidents (MCIs) has recently received an increasing amount of attention in the trauma and emergency medicine literature. The aim of this literature is to guide hospitals, in particular emergency departments (ED) and trauma centers, in the development of disaster response plans and protocols. Mass casualty incidents are typically accompanied by a large surge of victims with potentially life-threatening injuries requiring medical attention (Hirshberg et al. 1999). As a special form of MCI, mass casualty terrorist bombings (MCTBs) have been the particular focus of a growing subset of the research on MCI response planning. In addition to the increasing prevalence of terrorist bombings in recent years (Global Terrorism Database 2011), there is ample evidence showing that the victims of terrorist bombings present a specific injury pattern that distinguishes them from conventional trauma patients (Kluger 2003, Kluger et al. 2004). This distinct injury pattern validates the need to research the medical response to MCTBs as a unique problem within the wider problem of traditional MCI response planning.

Frykberg and Tepas (1988) published one of the earliest analyses of injury patterns observed after terrorist bombings, and their results have guided nearly all subsequent research published in this area. Their initial work compiles data from existing studies on 220 incidents occurring between 1969 and 1983, and a follow up study by Frykberg (2002) adds data from several prominent incidents between 1986 and 2001. These studies and others suggest four fundamental aspects of the medical response that must be considered in the development of a response plan: (1) objective of the medical response, (2) description and knowledge of the typical injury patterns, (3) triage, i.e. determining which victims are in need of immediate care, and (4) delivery of medical care.

5.1. Objectives, Injury Patterns, and Delivery of Care

Objective of the Medical Response

Most would agree that the general goal of the medical response to a MCI (terrorist bombing or otherwise) is to "save as many lives as possible," but there is no such consensus on exactly how this goal should translate to objectives for the delivery of medical care. The literature on this point falls into two main camps, those who advocate shifting focus from "doing the greatest good for each individual to doing the greatest good for the greatest number of people" (Armstrong et al. 2008, Frykberg 2002, Frykberg 2005) and those who advocate providing severely wounded victims with "a level of care that approximates the level of care provided to similar trauma patients under normal circumstances" (Ashkenazi et al. 2008, Hirschberg et al. 2001, Hirshberg et al. 2005). On the surface, the extent to which these two goals are contradictory is not immediately clear, but the difference becomes apparent when considering the treatment of the most severely wounded victims (who are not likely to survive even with optimal medical care).

The vagueness of these objectives makes them difficult to measure. However, some metrics have been proposed that allow health care providers and researchers to judge the efficacy of the medical response to an MCI. The *critical mortality rate* measures the percentage of severely wounded patients who die after their medical care has begun (Frykberg 1988, Frykberg 2004). This metric more closely aligns with the first objective above. The other metric, which aligns more closely with the second objective, focuses on the *surge capacity* of a hospital, or the hospital's capacity to treat newly-arriving severely wounded victims without degradation in the care they receive (Kosashvili et al. 2009, Peleg and Kellerman 2009, Rothman et al. 2006). The recent trend in the literature states surge capacity as an arrival rate and requires a method for measuring quality of care (Aylwin et al. 2006, Hirshberg et al. 2001, Hirschberg et al. 2005). In a recent simulation model, Hirshberg et al. (2005) determine the quality of care by assigning relative scores to the trauma teams (based on their compositions) assembled for severely injured patients.

Patterns of Injury Severity

While the magnitude of a MCI or MCTB is often judged publicly by the number of immediate casualties, the scope of the problem facing an ED or trauma center in the aftermath of such an event depends not on the immediate mortality rate but on the number of injured victims who arrive at the hospital seeking treatment (Hirshberg et al. 2001). Nearly all analyses of terrorist bombing injury severity patterns show that between 10 and 20 percent of injured survivors are severely wounded (Almogy et al. 2006, Aschkenasy-Steuer et al. 2005, Ashkenazi et al. 2008, Aylwin et al. 2006, Frykberg and Tepas 1988, Frykberg 2002, Kluger et al. 2004, and Turégano-Fuentes et al. 2008).

Several factors relating to the circumstances of the attacks have been shown to correlate with injury severity patterns. The number of immediate deaths is not surprisingly tied to the magnitude of the explosion, but other key factors include whether the bomb was detonated in an outdoor space or in a confined area and the collapse of any buildings or other structures (Frykberg 2002). Of these factors, the difference between open and confined spaces is the most significant in determining the rate of severely injured victims among the immediate survivors, with explosions taking place in confined spaces (such as buses, restaurants, and crowded marketplaces) leading to higher numbers of severely wounded (Frykberg and Tepas 1988, Frykberg 2002).

Delivery of Care

Before appropriate care can be delivered to the victims of an MCI, the casualties must go through triage, which is the process of determining which victims are in need of immediate care (Frykberg 2005, Halpern et al. 2003). We will discuss the triage process, which is itself the focus of a large volume of research, in the next section. In the meantime, after victims have been triaged, effectively delivering the appropriate medical care requires detailed plans for the flow of each patient category through the ED or trauma center. Typically the treatment area for those requiring immediate care is separated from the treatment area for those whose care can be delayed, and emphasis is placed on maintaining unidirectional patient flow (Ashkenazi et al. 2008,

Frykberg 2002, Hirshberg et al. 1999, Hirshberg et al. 2005). In order to prepare the hospital for the expected influx of casualties, patients in the ED should be transferred to hospital floor beds, all nonurgent activity (including scheduled elective surgeries) should be halted, and available personnel should be summoned (Almogy et al. 2004, Aschkenasy-Steuer et al. 2005). Some initial estimate of the number and severity of expected casualties is critical at this stage, and it is important that management have the flexibility to transform other beds to trauma and ICU beds in order to handle the surge (Almogy et al. 2004, Aschkenasy-Steuer et al. 2005, Aylwin et al. 2006, Turégano-Fuentes et al. 2008). However, Turégano-Fuentes et al. (2008) found that too many physicians, nurses, and students were called to the hospital, which crowded the ED, and that the decision to discharge all existing patients created more open beds than were actually needed. This suggests that while flexibility is important, overcompensating for the expected surge can also be detrimental to the effective delivery of care.

Once patient flow has been determined and the appropriate resources have been mobilized, guidelines must be in place for assigning medical providers to treatment areas and for the standard of care that should be given. Several authors advocate assigning specifically designed treatment teams (for instance, one attending physician and two residents) to designated trauma rooms or groups of ED beds (Almogy et al. 2004, Einav et al. 2004, Frykberg 2002). Ashkenazi et al. (2008) suggest that expert trauma surgeons, ICU staff, and anesthesiologists not be assigned to specific sites, but rather be free to readily assist in the treatment of those who require expert care. Multiple authors advocate for restricted radiology and laboratory testing and minimal blood bank usage while casualties are still arriving and the full scope of the incident is unknown (Frykberg 2002, Frykberg 2005, Hirshberg et al. 1999). During this initial phase, surgery should be limited and should focus on damage control (Almogy et al. 2004, Aylwin et al. 2006, Turégano-Fuentes et al. 2008). In contrast, Ashkenazi et al. (2008) argue that after the severely wounded patients are identified, all needed resources should be allocated to their treatment and should not be delayed until the scope of the incident is known. Finally, Hirshberg et al. (2001) advocate that non-

critically wounded patients be treated using the principle of minimal acceptable care in order to preserve trauma resources.

5.2. Triage

Both the objectives discussed above and the strategies for delivering appropriate medical care focus the primary efforts of the trauma response to a MCI on the effective delivery of life-saving care to those immediate survivors identified as severely wounded. For this reason the triage process, which serves to separate the severely wounded from the other immediate survivors, is perhaps the most-discussed component of MCI response planning. Varying opinions can be found in the literature on nearly every aspect of the triage process, including how and where it should be conducted, who should conduct it, and how accurate it must be to meet the overall objective. Armstrong et al. (2008) claim that the characteristics of an effective triage plan are simplicity, time efficiency, predictive validity, reliability, and accuracy, and all of the opinions on triage discussed below aim to improve the process in some or all of these aspects.

Categories

In general, there are five triage categories based on the level of treatment the victim requires: (1) those who require immediate care, (2) those whose care may be safely delayed, (3) those who require minimal care, often referred to as the walking wounded, (4) those who are so severely wounded that they are unlikely to live even with medical care, referred to as expectant, and (5) those who are dead (Frykberg 2002, Frykberg 2005, Lerner et al. 2008). However, in the context of an MCI, the most important distinction for in-hospital triage is between those who require immediate care and those who do not (Aylwin et al. 2006, Frykberg 2005, Hirshberg et al. 2001). Aylwin et al. (2006) also argue for using two categories during on-site triage for those who require transport and those who do not.

Complicating the separation of patients into those who require immediate care and those whose care can wait is the reality that not all patients who receive treatment will ultimately survive. There is consensus in the literature that triage decisions

should incorporate the impact of treatment on long-term (post-treatment) patient survival (Frykberg 2005, Lerner et al. 2008). Furthermore, the widely accepted trauma concept of the "golden hour" suggests that patients' long-term survival probabilities deteriorate as they wait for treatment (Lerner and Moscati 2001, Sacco 2005).

This notion of long-term survival probability, and its possible deterioration, is closely related to the expectant category mentioned above, which happens to be a particularly controversial aspect of MCI triage. Outside the context of a MCI, every effort would be made to treat expectant patients in spite of the low probability that treatment will save their lives. However, in the aftermath of a MCI, and in keeping with the stated goal of doing the greatest good for the greatest number of people, Frykberg and others claim that patients in the expectant category should not be treated (outside of palliative care) because critical resources can be better used on patients who are more likely to benefit from treatment (Almogy et al. 2004, Frykberg and Tepas 1988, Frykberg 2002, Frykberg 2005). Even Hirshberg, who advocates giving severely wounded victims a level of care approximating the care they would receive under normal circumstances, acknowledges the need to focus treatment on patients that are most likely to benefit from treatment (Hirshberg 2004). Only Ashkenazi et al. (2008) refuse to accept an expectant category, and argue that it is unacceptable to deny these patients care just because their injuries occur during a MCI.

How, Where, When, and Who?

Of course, lost in the discussion of how to deal with victims in the expectant category is the need for effective tools to actually separate patients into these categories. Both Lerner et al. (2008) and Jenkins et al. (2008) provide a survey of commonly used triage algorithms, but both papers acknowledge that there is little scientific evidence to support one algorithm over another. Thoroughly reviewing the wealth of research in this area is outside the scope of this work, but some key findings are worth mentioning because of their ability to simplify and accelerate the triage process when faced with a large surge of MCTB victims. Meredith et al. (1995)

suggest that simplified triage focusing on the ability to follow verbal commands is highly accurate in predicting which patients require urgent trauma care. Other research finds that basing triage specifically on physiologic and anatomic indicators (ignoring mechanism of injury) significantly improves triage accuracy without adversely affecting outcomes (Cook et al. 2001). In the specific context of a terrorist bombing, triage can capitalize on the distinct injury pattern associated with these incidents. Specific external signs have been shown to accurately predict two of the leading causes of critical mortality after MCTBs, suggesting that more efficient triage could be achieved after a terrorist bombing by focusing exclusively on external signs (Almogy et al. 2006).

Some level of triage is always required at the scene of the incident to separate the immediate survivors from the dead, but after this initial assessment the remainder of the triage process focuses on classifying the survivors. Several studies argue that primary triage should occur on- or near-site followed by secondary triage at the entrance to the hospital (Aylwin et al. 2006, Frykberg and Tepas 1988, Frykberg 2002, Frykberg 2005, Turégano-Fuentes et al. 2008). Aschkenasy-Steuer et al. (2005) claim that pre-hospital triage cannot be trusted because not all wounded victims are transported to the hospital by emergency medical services (for instance, the walking wounded often take themselves to the hospital). Other authors argue that patients initially triaged as not requiring immediate treatment be regularly reassessed after being admitted to lower-intensity areas of the ED in order to avoid triage errors (Almogy et al. 2004, Armstrong et al. 2008, Aschkenasy-Steuer et al. 2005, Hirshberg 2004). However, another recent study points to empirical evidence suggesting that severe injuries are not often missed during primary triage, which makes repeated triage an unnecessary use of valuable resources (Ashkenazi et al. 2008).

In a recent paper, Armstrong et al. (2008) admit that the question of "who should perform mass casualty triage across settings and how these multidisciplinary professionals should be trained as triage officers" is understudied and remains "ripe for investigation". Aylwin et al. (2006) report that on-scene triage performed by trained EMS was more accurate than that performed by ambulance services and medically-trained bystanders. The conventional wisdom says that in-hospital triage

should be performed by an experience trauma, emergency medicine, or general surgeon (Almogy et al. 2004, Frykberg 2005, Kluger 2003, Turégano-Fuentes et al. 2008). However, the results of a simulation model of the response to a terrorist bombing indicated that the accuracy of triage has little impact on outcomes, which suggests that triage need not be performed by the most experienced surgeons (Hirshberg et al. 1999). This view is echoed by Ashkenazi et al. (2008), who argue that because the most important asset for patient survival is an experience trauma surgeon, this individual should not be wasted on triage.

Accuracy and Situational Awareness

Behind each of these how, where, and who opinions on triage lies some understanding of the importance of triage accuracy. Armstrong et al. (2008) point out that triage is essentially a form of communication, and the lack of reliable communication is known to be a harbinger of poor outcomes in disaster management. Triage accuracy is defined in terms of over-triage (classifying a non-severely wounded patient as severely wounded) and under-triage (classifying a severely wounded patient as non-severely wounded). Under-triage is clearly the more life-threatening of these two alternatives, and for this reason over-triage rates as high as 50 percent are often accepted as necessary during MCTBs to limit under-triage (Cook et al. 2001, Frykberg 2002, Kluger 2003). Several analyses by Frykberg and co-authors confirm this notion, finding that under-triage is usually negligible (less than 1 percent) while over-triage averages over 50 percent (Frykberg and Tepas 1988, Frykberg 2002, Frykberg 2005). These same studies show that over-triage rates are positively correlated with critical mortality rates, suggesting that treating too many patients as severely wounded bogs down critical trauma resources. A different analysis of triage decisions at a hospital in Israel, on the other hand, found a significant rate of under-triage, indicating that under-triage is not negligible in all cases (Ashkenazi et al. 2006).

Contrasting some of these other findings, Hirshberg et al. (1999) found in a simulation study that reducing the over-triage rate did not have a significant impact on outcomes. Other authors similarly argue that the negative impact of triage errors

can be mitigated by effective planning and flexibility. Another simulation study concluded that the ratio of critically injured casualties to available treatment units affects critical mortality more than over-triage rates (Hupert et al. 2007). Aylwin et al. (2006) found that high over-triage was not associated with high critical mortality in one high profile MCTB. Others note that over-triage can be compensated for by the use of improvised trauma beds, while the repeated reassessment of non-severely injured patients can catch victims with delayed presentation of severe wounds (Ashkenazi et al. 2008, Turégano-Fuentes et al. 2008).

Lastly, there is broad consensus that triage should be performed with a high degree of situational awareness. Almogy et al. (2004) suggest that rough information regarding the number of casualties and physical location of the incident should be communicated to the hospital as soon as possible, to which Armstrong et al. (2008) add that "the application of triage in mass casualties varies by casualty load and resource availability". Acknowledging the controversy involved in not treating the expectant category, the bounds of this category should differ by incident type and location, and should be determined based on numbers and types of casualties and resource availability (Armstrong et al. 2008, Frykberg 2002, Frykberg 2005).

5.3. Mathematical Modeling and a Direction for Research

The vast majority of the recommendations and opinions advocated in the papers referenced above are based on data analysis of past incidents and the personal experiences of the authors in managing the medical response to MCIs and terrorist bombings. Just as the push for evidence-based decision-making has gained momentum in other areas of medical research, mathematical modeling has slowly emerged over the last decade or so as a useful tool in the development of MCI response plans. In particular, simulation studies of different components of the response have become increasingly common. Christie and Levary (1998) simulate the transportation of seriously injured casualties to nearby hospitals in the wake of an MCI and explore the relationships between resource availability, casualty loads, proximity to hospitals, and patient waiting times. As discussed above, Hirshberg et al. (1999), Hirshberg et al. (2005), and Hupert et al. (2007) simulate the in-hospital

response to a MCI. These papers largely focus on the relationships between triage accuracy, resource availability, and common metrics such as the critical mortality rate and surge capacity.

A shortcoming of these simulation studies, particularly as they relate to the many triage-related questions raised above, is their simplification of the triage decision itself. In the three papers that model the in-hospital response, triage (the classification of a patient as requiring immediate or delayed care) is modeled by identifying a fixed percentage of all arriving casualties as severely wounded. Triage inaccuracy is modeled similarly, with some fixed percentage of initial triage classifications being incorrect. In reality, as suggested by the medical research reviewed above, triage decisions must take into account a number of dynamic factors, including the total number of casualties, the availability of resources, and a range of patient characteristics (most triage algorithms use at least three measures of injury severity).

Two recent papers by Sacco et al. (2005) and Argon et al. (2008) address the triage decision more directly. The Sacco Triage Method (STM) is a resource-constrained linear programming model that optimizes on-scene triage in the face of limited on-scene and transportation resources. Both the initial injury severity and long-term survival probabilities of patients are estimated, and these estimates are used to make triage decisions that maximize the total expected number of survivors. Argon et al. (2008) approach the MCI triage problem as an impatient jobs clearing system, where a large number of patients is generated almost simultaneously and must be scheduled for treatment. Each patient is associated with a random treatment time and a random length of time that he or she is capable of surviving while waiting for treatment (a measure of impatience), and the model optimizes the number of patients that eventually receive treatment. The results of both these studies confirm the notion that MCI triage decisions must be made dynamically, taking into account patient characteristics and resource constraints, in order to truly maximize the number of lives saved.

In the following chapter, we propose an extension of the model studied by Argon et al. (2008) that adds an important third dimension to the patient characteristics incorporated in their model. In particular, we add the deteriorating long-term (post-

treatment) survival probabilities used in the Sacco Triage Method (and suggested by the "golden hour" principle) to the treatment times and pre-treatment survival times already used by Argon et al. The resulting model can be described as an impatient jobs with diminishing rewards clearing system, and our subsequent analysis focuses on the impact of deteriorating long-term survival probabilities on the resulting MCI triage decisions.

Chapter 6. Mass Casualty Incident Triage: The Impatient Jobs with Diminishing Rewards Clearing System

6.1. Background and Motivation

As the previous chapter discusses, the development of mass casualty incident (MCI) response plans and protocols has become an important priority for public health and hospital administrators in recent years, and the management of the medical response to MCIs has correspondingly received an increasing amount of attention in the trauma and emergency medicine literature. Triage plays an integral role in the success of the response to a MCI by identifying which patients are to receive immediate care, but the literature shows that there are still many unresolved questions surrounding the design and implementation of the triage process. Triage algorithms used in practice categorize surviving patients into one of four groups based on their *need* for immediate treatment, reflecting the fact that patients may die while waiting for treatment (Lerner et al. 2008). The volume of casualties generated by a MCI forces triage algorithms to also consider whether patients will *benefit* from immediate treatment, since not all patients receiving treatment will ultimately survive (Frykberg 2005). Exactly how the likelihood of survival after treatment should be incorporated into triage decisions remains a subject of much debate.

The research by Argon et al. (2008) mentioned in Chapter 5 models the triage process as a single-server clearing system with impatient jobs (patients) that will abandon the system (die) if forced to wait too long for service (treatment). That work is one in a series of recent papers on scheduling impatient jobs in a clearing system (Argon et al. 2008, Glazebrook et al. 2004, Li and Glazebrook 2010). Glazebrook et al. (2004) associate different rewards with different jobs and explore static policies aimed at maximizing the total reward earned by the server. Both Argon et al. (2008) and Li and Glazebrook (2010) assume that all jobs have the same reward and use stochastic dynamic programming-based, state-dependent heuristics to maximize the

number of jobs that receive service. These dynamic heuristics outperform traditional static heuristics, suggesting that patient triage in the aftermath of a MCI should rely on state-dependent policies that take into account factors such as the number of casualties and resource availability. Before discussing our extension of the impatient jobs clearing system, we note that the broader literature on stochastic job scheduling in queuing and clearing systems with abandonment extends well beyond these three articles (see, for instance, Boots and Tijms 1999, Iravani and Balcıoğlu 2008, Zhao et al. 1991). However, these papers by Glazebrook et al., Argon et al. and Li and Glazebrook represent, to the best of our knowledge, the only research that considers the "MCI triage" clearing system.

As mentioned earlier, a fundamental component of MCI triage, and of trauma medicine in general, is the reality that not all patients who receive treatment will ultimately survive. Furthermore, patients' long-term survival probabilities may deteriorate the longer they wait for treatment. This concept is absent from the papers by Argon et al. (2008) and Li and Glazebrook (2010), but can be incorporated into their impatient jobs clearing system model in the form of rewards for service that diminish as jobs are forced to wait. Glazebrook et al. (2004) consider constant rewards, but not the case with diminishing rewards. The objective, in the triage context, of the resulting impatient jobs with diminishing rewards clearing system is to maximize the expected number of survivors (the sum of the survival probabilities of those patients who are treated at the time they are treated), rather than the number of patients treated. The resulting problem is a generalization of the problems analyzed by Glazebrook et al. (2004), Argon et al. (2008), and Li and Glazebrook (2010) that we feel more accurately captures the realities of MCI triage. The goal of our research is to explore the impact of deteriorating survival probabilities on MCI triage decisions through the use of dynamic, state-dependent heuristics.

Before continuing with our formulation and analysis, it is important to acknowledge some of the assumptions and limitations associated with the single-server model. In practice, triage is the entry point into a large and complex healthcare delivery system that consists of many treatment locations with separate queues and different processing times (see, for instance, Hirshberg et al. 2005 and Hupert et al.

2007). In modeling MCI triage as a single-server system, our server represents the post-triage delivery system as a whole and our service times model the departure process from this system rather than the time required for a particular medical procedure. This interpretation of service times as inter-departure times suggests that care will need to be taken when extending our model to include multiple servers.

The remainder of the chapter will be laid out as follows. Section 6.2 presents a general formulation of the single-server impatient jobs "MCI triage" clearing system, as well as a formulation for a special Markov case that will be used in the development of heuristic policies. In Section 6.3 we review the heuristics proposed in the existing literature, show how they can be extended to the problem with diminishing rewards, and propose new heuristics. Section 6.4 discusses our use of the medical literature on MCIs and MCI triage to generate a set of test problems. The resulting set of problems is used in Section 6.5 to test the performance of the proposed heuristics, and Section 6.6 offers a discussion of our results.

6.2. Impatient Jobs with Diminishing Rewards

The problem of scheduling impatient jobs in a single-server clearing system with diminishing rewards can be formally stated as follows. A single server is tasked with providing service to a finite collection of simultaneously arriving jobs. Each job is associated with a random service time and a random lifetime, with the latter representing how long the job will wait for service before abandoning the system. The server receives a positive reward for each job that is serviced. The magnitude of these rewards differs between jobs and may decrease as a function of the time that service is initiated. The server's objective is to sequence the collection of jobs to maximize the expected reward earned, stopping only when all jobs have either been served or abandoned the system.

General Formulation

A stochastic dynamic programming (SDP) formulation for the problem can be obtained by extending the formulation given in Li and Glazebrook (2010). We assume that the jobs are divided into K classes, where class j is identified by a unique

combination of service time (X_j with distribution function F_j) and lifetime (Y_j with distribution function G_j) random variables and a non-increasing, positive reward function ($R_j(t)$). Let N_j be the number of jobs of class j present at time 0. We assume that the service and lifetime random variables, respectively, are i.i.d. within classes and independent between classes. The state of the system at each decision epoch t (at time 0 and at the conclusion of a job's service) can be described by the number of jobs $\boldsymbol{n} = (n_1, n_2, \ldots, n_K)$ of each class remaining in the system. We define the value function $V(\boldsymbol{n}, t)$ as the maximum expected rewards earned for being in state $\boldsymbol{n}$ at decision epoch t. If a job of class j is taken into service at decision epoch t, the system evolves in the following manner. Let s represent the length of the selected job's service (a realization of its service time random variable) and $\boldsymbol{n}' = (n_1', n_2', \ldots, n_K')$ represent the state of the system at the end of the job's service (the next decision epoch at time $t' = t + s$). The number of abandonments from job class i (due to expired lifetimes) during class j's service can be expressed as $n_i - n_i' - \delta_{ij}$ (where δ_{ij} equals one when $i = j$ and equals zero otherwise). Let $p(\boldsymbol{n}'|\boldsymbol{n}, t, j, s)$ be the probability of transitioning from state $\boldsymbol{n}$ at decision epoch t to state $\boldsymbol{n}'$ at decision epoch $t' = t + s$ during the service of a job from class j. The optimality equation and boundary condition for the SDP (formulation (1)) can be expressed as follows.

$$V(\boldsymbol{n}, t) = \max_{j \mid n_j > 0} \{R_j(t) + E[V(\boldsymbol{n}', t')]\}$$

$$= \max_{j \mid n_j > 0} \left\{ R_j(t) + \int_0^\infty \sum_{\boldsymbol{n}'} p(\boldsymbol{n}'|\boldsymbol{n}, t, j, s)\, V(\boldsymbol{n}', t+s)\, dF_j(s) \right\}, \text{for } \boldsymbol{n} \neq \boldsymbol{0}$$

$$V(\boldsymbol{0}, t) = 0$$

The only difference between this formulation and the one given in Li and Glazebrook (2010) is the inclusion of the reward function $R_j(t)$ in the optimality equation (as opposed to the constant, uniform rewards studied in their paper). This slight difference introduces significant difficulty from a computational standpoint because it ensures that in general the time dimension must be included in the state space. As a result, our analysis of this problem focuses on the development of heuristic procedures. One approach for developing heuristics is to exploit structural

behavior in the optimal policies for problems that can be efficiently solved to optimality. However, strong assumptions are required on the distributions of the random variables and the shape of the reward functions in order to be able to efficiently find optimal policies. The next subsection presents an alternative formulation for one such set of assumptions, namely that the random variables are exponentially distributed (the Markov case) and the reward functions are exponentially decaying to zero at a uniform rate.

The Markov Case with Uniformly Decaying Rewards

We assume that jobs are still divided into classes with associated service and lifetime random variables. We make the additional assumptions that the random variables are exponentially distributed and that the rewards for service decay exponentially to zero at a uniform rate. The following formulation is an extension of the formulation given by Argon et al. (2008) for the Markov case with uniform rewards. Let μ_j^{-1} and r_j^{-1} be the mean service time and lifetime, respectively, for a job in class j (i.e., $E[X_j] = \mu_j^{-1}$ and $E[Y_j] = r_j^{-1}$). Let the reward function for jobs from class j be given by $R_j(t) = R_j e^{-\lambda t}$ for some constant $\lambda > 0$. The state of the system at time t can be described by the number of jobs of each class in the system (including the job in service), $\boldsymbol{n}$, and the state of the server, $Q \epsilon \{E, P_1, \dots, P_k\}$. In defining the state of the server, E indicates that the server is empty and P_j indicates that the server is busy with a job of type j. Decisions can only occur when the server is empty. Note that the system can only be instantaneously empty because any remaining jobs will always be available and the next job chosen will enter service immediately. Transitions between states occur either when a job is taken into service or when a service time or lifetime event occurs. We can take advantage of the exponential service time and lifetime distributions to characterize the distribution of the time until the next transition and to determine the probability associated with each of the possible transitions. Define $A_j(\boldsymbol{n})$ be defined as the subset of job classes that have jobs in the queue when the system is in state $(\boldsymbol{n}, P_j, t)$ (i.e. the subset of job classes such that $n_i - \delta_{ij} > 0$). Define $\nu_j^{\boldsymbol{n}} = \mu_j + \sum_{i \in A_j(\boldsymbol{n})} (n_i - \delta_{ij}) r_i$ as the rate at

which the next event will occur if the system is in state $(\boldsymbol{n}, P_j, t)$. We will use the notation $\boldsymbol{e}_i$ to refer to the unit vector in the ith direction. The value function $\tilde{V}(\boldsymbol{n}, Q, t)$ for this alternative formulation is defined as the maximum expected rewards earned from being in state $(\boldsymbol{n}, Q, t)$. The optimality equations and boundary conditions for this formulation can then be stated as follows.

$$\tilde{V}(\boldsymbol{n}, E, t) = \max_{j|n_j>0}\{R_j(t) + \tilde{V}(\boldsymbol{n}, P_j, t)\}, \text{for } \boldsymbol{n} \neq \mathbf{0}$$

$$\tilde{V}(\mathbf{0}, E, t) = 0$$

$$\tilde{V}(\boldsymbol{n}, P_j, t) = \int_0^\infty \nu_j^{\boldsymbol{n}} e^{-\nu_j^{\boldsymbol{n}} s}\left[\frac{\mu_j}{\nu_j^{\boldsymbol{n}}}\tilde{V}(\boldsymbol{n} - \boldsymbol{e_j}, E, t+s) + \sum_{i\in A_j(\boldsymbol{n})}\frac{(n_i - \delta_{ij})r_i}{\nu_j^{\boldsymbol{n}}}\tilde{V}(\boldsymbol{n} - \boldsymbol{e_i}, P_j, t+s)\right] ds, \text{for } n_j > 0$$

$$\tilde{V}(\boldsymbol{n}, P_j, t) = 0, \text{if } n_j = 0$$

We will use the following lemma to show that the time dimension can be effectively removed from this formulation, taking advantage of the assumption that the reward functions decay exponentially to zero at a uniform rate.

Lemma 6.1 *If the service time and lifetime random variables are exponentially distributed with rates μ_j and r_j, respectively, and the reward functions are given by $R_j(t) = R_j e^{-\lambda t}$ for some constant $\lambda > 0$, then $\tilde{V}(\boldsymbol{n}, Q, t) = e^{-\lambda t}\tilde{V}(\boldsymbol{n}, Q, 0)$ for all states $(\boldsymbol{n}, Q, t)$.*

Proof:

The proof proceeds by induction on the number of jobs in the system, $\boldsymbol{n}$. The base case for $\boldsymbol{n} = \mathbf{0}$ holds trivially. Assume that the claim holds for all states $\boldsymbol{n}'$ such that $n_i' \leq n_i \ \forall i$ and $n_i' < n_i$ for at least one i. Note that this assumption implies that for all such states $\boldsymbol{n}'$, $\tilde{V}(\boldsymbol{n}', Q, t+s) = e^{-\lambda t}\tilde{V}(\boldsymbol{n}', Q, s)$.

$$\tilde{V}(\boldsymbol{n}, P_j, t) = \int_0^{\infty} \nu_j^{\boldsymbol{n}} e^{-\nu_j^{\boldsymbol{n}} s} \left[\frac{\mu_j}{\nu_j^{\boldsymbol{n}}} \tilde{V}(\boldsymbol{n} - \boldsymbol{e_j}, E, t+s) + \sum_{i \in A_j(\boldsymbol{n})} \frac{(n_i - \delta_{ij}) r_i}{\nu_j^{\boldsymbol{n}}} \tilde{V}(\boldsymbol{n} - \boldsymbol{e_i}, P_j, t+s) \right] ds$$

$$= \int_0^{\infty} \nu_j^{\boldsymbol{n}} e^{-\nu_j^{\boldsymbol{n}} s} \left[\frac{\mu_j}{\nu_j^{\boldsymbol{n}}} e^{-\lambda t} \tilde{V}(\boldsymbol{n} - \boldsymbol{e_j}, E, s) + \sum_{i \in A_j(\boldsymbol{n})} \frac{(n_i - \delta_{ij}) r_i}{\nu_j^{\boldsymbol{n}}} e^{-\lambda t} \tilde{V}(\boldsymbol{n} - \boldsymbol{e_i}, P_j, s) \right] ds$$

$$= e^{-\lambda t} \int_0^{\infty} \nu_j^{\boldsymbol{n}} e^{-\nu_j^{\boldsymbol{n}} s} \left[\frac{\mu_j}{\nu_j^{\boldsymbol{n}}} \tilde{V}(\boldsymbol{n} - \boldsymbol{e_j}, E, s) + \sum_{i \in A_j(\boldsymbol{n})} \frac{(n_i - \delta_{ij}) r_i}{\nu_j^{\boldsymbol{n}}} \tilde{V}(\boldsymbol{n} - \boldsymbol{e_i}, P_j, s) \right] ds$$

$$= e^{-\lambda t} \tilde{V}(\boldsymbol{n}, P_j, 0)$$

$$\tilde{V}(\boldsymbol{n}, E, t) = \max_{j | n_j > 0} \{ R_j(t) + \tilde{V}(\boldsymbol{n}, P_j, t) \}$$

$$= \max_{j | n_j > 0} \{ R_j e^{-\lambda t} + e^{-\lambda t} \tilde{V}(\boldsymbol{n}, P_j, 0) \}$$

$$= e^{-\lambda t} \max_{j | n_j > 0} \{ R_j + \tilde{V}(\boldsymbol{n}, P_j, 0) \}$$

$$= e^{-\lambda t} \tilde{V}(\boldsymbol{n}', E, 0)$$

■

Using this lemma it is clear to see that the optimal decision in state $(\boldsymbol{n}, E, t)$ will be identical to the optimal decision in state $(\boldsymbol{n}, E, 0)$. The following re-statement of the optimality equations demonstrates that all optimal decisions can be determined based on the value of states at time 0, implying that the time dimension can essentially be ignored when solving the SDP. We shall refer to this re-formulation as formulation (2).

$$\tilde{V}(\boldsymbol{n}, E, 0) = \max_{j|n_j>0}\{R_j + \tilde{V}(\boldsymbol{n}, P_j, 0)\}, \text{ for } \boldsymbol{n} \neq \boldsymbol{0}$$

$$\tilde{V}(\boldsymbol{n}, P_j, 0) = \int_0^\infty \nu_j^{\boldsymbol{n}} e^{-\nu_j^{\boldsymbol{n}} s}\left[\frac{\mu_j}{\nu_j^{\boldsymbol{n}}}\tilde{V}(\boldsymbol{n}-\boldsymbol{e_j}, E, s) + \sum_{i\in A_j(\boldsymbol{n})}\frac{(n_i-\delta_{ij})r_i}{\nu_j^{\boldsymbol{n}}}\tilde{V}(\boldsymbol{n}-\boldsymbol{e_i}, P_j, s)\right]ds$$

$$= \int_0^\infty e^{-\nu_j^{\boldsymbol{n}} s}\left[\mu_j e^{-\lambda s}\tilde{V}(\boldsymbol{n}-\boldsymbol{e_j}, E, 0) + \sum_{i\in A_j(\boldsymbol{n})}(n_i-\delta_{ij})r_i e^{-\lambda s}\tilde{V}(\boldsymbol{n}-\boldsymbol{e_i}, P_j, 0)\right]ds$$

$$= \int_0^\infty e^{-(\nu_j^{\boldsymbol{n}}+\lambda)s}ds\left[\mu_j\tilde{V}(\boldsymbol{n}-\boldsymbol{e_j}, E, 0) + \sum_{i\in A_j(\boldsymbol{n})}(n_i-\delta_{ij})r_i\tilde{V}(\boldsymbol{n}-\boldsymbol{e_i}, P_j, 0)\right]$$

$$= \frac{\mu_j}{\nu_j^{\boldsymbol{n}}+\lambda}\tilde{V}(\boldsymbol{n}-\boldsymbol{e_j}, E, 0) + \sum_{i\in A_j(\boldsymbol{n})}\frac{(n_i-\delta_{ij})r_i}{\nu_j^{\boldsymbol{n}}+\lambda}\tilde{V}(\boldsymbol{n}-\boldsymbol{e_i}, P_j, 0), \text{ for } n_j > 0$$

The resulting formulation, which is expressed exclusively using states at time 0, differs from the formulation in Argon et al. (2008) in the inclusion of the different initial reward values ($R_j(0) = R_j$) and the rate at which the rewards are decaying (λ). It is important to note that Lemma 6.1 (and thus this re-formulation) does not apply if the rewards do not decay at the same rate for all job classes, or do not decay to zero. We will use formulation (2) to extend the structural results presented by Argon et al., which in turn illustrate how their heuristics can be extended to the problem with diminishing rewards.

6.3. Scheduling Heuristics

In developing heuristics for the single-server impatient jobs clearing system with diminishing rewards, it is logical to first extend the heuristics published in the existing literature on special cases of the problem. The heuristics proposed by Glazebrook et al. (2004), Argon et al. (2008), and Li and Glazebrook (2010) differ

markedly in their approaches, and the methods required to extend them vary accordingly. In addition to these extensions, we present a new heuristic which is different in approach but similar in feel to Argon et al.'s rule.

Our heuristics for the single-server impatient jobs clearing system (with or without diminishing rewards) are similar to dispatching rules commonly used in manufacturing scheduling. At each decision epoch, all remaining job classes are sequenced according to some priority rule and a job from the highest priority class is selected for service. As a result, the statement of a heuristic equates to presenting the rule used to compute priorities for each job class. Each of the heuristics we present will be dynamic (rather than static), meaning that priorities are recomputed and the job classes are reordered at each decision epoch (rather than using a fixed ordering determined at time 0).

In the following discussion, we will continue to consider the Markov case where all random variables are exponentially distributed with $E[X_j] = \mu_j^{-1}$ and $E[Y_j] = r_j^{-1}$. Specific assumptions on the shape of the reward functions will be made when appropriate. For the heuristics that rely on the Markov assumption, we will also discuss how they can be further generalized and applied to the problem with general lifetime and service time distributions and reward functions.

$Rr\mu$ Rule

Glazebrook et al. (2004) consider constant rewards ($R_j(t) = R_j$) and propose a static $Rr\mu$ rule that is a variant of the $c\mu$ rule frequently seen in the job scheduling literature (see, for instance, Van Mieghem 1995). At each decision epoch, the $Rr\mu$ rule chooses from among the available jobs according to which job class has the maximum value of $R_j r_j \mu_j$. A natural extension of the $Rr\mu$ that takes into account the diminishing nature of the rewards simply replaces the constant R_j with the value of the reward function at each decision epoch. The result is a dynamic policy that chooses the job with the maximum value of $R_j(t) r_j \mu_j$. An obvious shortcoming of this first extension is that it only uses a snapshot of the reward values at each decision epoch, without taking into account the rate at which the rewards are decreasing. This

could potentially become important in realistic scenarios where the rewards for different job classes decrease at different rates.

One way to incorporate the shape of the reward functions into the $Rr\mu$ rule is to realize the similarities between the diminishing reward functions and the impatience of the jobs. The decay of the reward functions and the expiration of the jobs' lifetimes both represent the departure of potential rewards from the system, which suggests a $R(r+\lambda)\mu$ rule that adds the rate at which the rewards are decreasing to the rate at which jobs are abandoning the system. For general reward functions, this λ can be thought of as the hazard rate associated with each of the reward functions. If the reward function for class j is assumed to be differentiable, then the hazard rate function can be defined as $\theta_j(t) = -R_j'(t)[R_j(t)]^{-1}$. The resulting $R(r+\lambda)\mu$ rule chooses the available job with the maximum value of $R_j(t)[r_j+\theta_j(t)]\mu_j$. Note that in the case discussed in formulation (2) where all rewards decay exponentially to zero at a uniform rate, $\theta_j(t) = \lambda$, which is simply the uniform rate at which all the rewards decay.

Triangular Rule

Argon et al. (2008) consider the Markov case with constant unit rewards ($R_j(t) = 1$) and explore the structure of optimal SDP policies for problems with two jobs classes under certain realistic restrictions (motivated by MCI triage) on the values of r_j and μ_j. Their proposed "triangular" heuristic (TRI) chooses the available job class j that minimizes the approximate expected number of jobs abandoning the system during its service, given by the following expression:

$$\frac{1}{\mu_j}\sum_{i\in A_j(\boldsymbol{n})}(n_i-\delta_{ij})r_i$$

For two jobs classes, this rule roughly defines a triangular region in the state space inside which the more time critical job class (smaller $E[Y_j]$, or larger μ_j) is selected and outside which the less time critical job class (larger $E[Y_j]$, or smaller μ_j) is selected. The following lemma extends one of the structural results from Argon et al.

to the more general Markov case considered above, where the rewards differ between classes but decay exponentially to zero at the same rate. The lemma is stated for two job classes, but the heuristic it motivates can be applied to problems with more than two classes.

Lemma 6.2 *Consider the problem with two job classes, exponentially distributed service times and lifetimes with rates μ_j and r_j, respectively, and reward functions given by $R_j(t) = R_j e^{-\lambda t}$ for some constant $\lambda > 0$. Assume that $r_1 < r_2 < \mu_2 < \mu_1$, $R_2 \leq R_1$, $n_1 > 1$, and $n_2 > 1$. If*

$$(i)\ R_2 + \tilde{V}(\boldsymbol{n} - \boldsymbol{e_i}, P_2, 0) \geq R_1 + \tilde{V}(\boldsymbol{n} - \boldsymbol{e_i}, P_1, 0) \text{ for } i = 1,2, \text{ and}$$

$$(ii)\ n_1 r_1 + n_2 r_2 \leq \frac{R_2(\mu_2 + \lambda)(\mu_1 - r_1 + \lambda) - R_1(\mu_1 + \lambda)(\mu_2 - r_2 + \lambda)}{R_1(\mu_1 + \lambda) - R_2(\mu_2 + \lambda)},$$

then for sufficiently small λ, $R_2 + \tilde{V}(\boldsymbol{n}, P_2, 0) \geq R_1 + \tilde{V}(\boldsymbol{n}, P_1, 0)$.

Proof:

We begin with an important claim based on a lemma in Argon et al. (2008), which will play a role in the remainder of the proof.

Claim: If $\mu_2 < \mu_1$, then $\tilde{V}(\boldsymbol{n} + \boldsymbol{e_2}, P_2, 0) < \tilde{V}(\boldsymbol{n} + \boldsymbol{e_1}, P_1, 0)$.

For the proof of this claim, please refer to the proof of Lemma 2 in Argon et al., which can easily be modified to incorporate rewards. Intuitively, the claim states that if the resulting number of jobs in the queue is identical ($\boldsymbol{n}$), then it is better to be serving a job with shorter service time. Also observe that this claim will be true for all values of λ, but especially for small values of λ (since the difference between $\tilde{V}(\boldsymbol{n} + \boldsymbol{e_2}, P_2, 0)$ and $\tilde{V}(\boldsymbol{n} + \boldsymbol{e_1}, P_1, 0)$ will decrease for larger values of λ that send the reward functions to zero more quickly).

The proof of the lemma requires us to show that under the specified conditions, and for sufficiently small λ, $[R_2 + \tilde{V}(\boldsymbol{n}, P_2, 0)] - [R_1 + \tilde{V}(\boldsymbol{n}, P_1, 0)] \geq 0$. We first note that condition (i) implies that $\tilde{V}(\boldsymbol{n} - \boldsymbol{e_2}, E, 0) = R_2 + \tilde{V}(\boldsymbol{n} - \boldsymbol{e_2}, P_2, 0)$ and

$\tilde{V}(\boldsymbol{n}-\boldsymbol{e_1},E,0)=R_2+\tilde{V}(\boldsymbol{n}-\boldsymbol{e_1},P_2,0)$ (and that the optimal decision in both states is to choose a job from class 2).

$$
\begin{aligned}
&[R_2+\tilde{V}(\boldsymbol{n},P_2,0)]-[R_1+\tilde{V}(\boldsymbol{n},P_1,0)]= \\
&\quad=\left[R_2+\frac{\mu_2}{\nu_2^{\boldsymbol{n}}+\lambda}\tilde{V}(\boldsymbol{n}-\boldsymbol{e_2},E,0)+\frac{n_1r_1}{\nu_2^{\boldsymbol{n}}+\lambda}\tilde{V}(\boldsymbol{n}-\boldsymbol{e_1},P_2,0)\right. \\
&\qquad\left.+\frac{(n_2-1)r_2}{\nu_2^{\boldsymbol{n}}+\lambda}\tilde{V}(\boldsymbol{n}-\boldsymbol{e_2},P_2,0)\right] \\
&\quad-\left[R_1+\frac{\mu_1}{\nu_1^{\boldsymbol{n}}+\lambda}\tilde{V}(\boldsymbol{n}-\boldsymbol{e_1},E,0)+\frac{(n_1-1)r_1}{\nu_1^{\boldsymbol{n}}+\lambda}\tilde{V}(\boldsymbol{n}-\boldsymbol{e_1},P_1,0)\right. \\
&\qquad\left.+\frac{n_2r_2}{\nu_1^{\boldsymbol{n}}+\lambda}\tilde{V}(\boldsymbol{n}-\boldsymbol{e_2},P_1,0)\right] \\
&\quad=\left[\frac{R_2(\nu_2^{\boldsymbol{n}}+\lambda)+\mu_2[R_2+\tilde{V}(\boldsymbol{n}-\boldsymbol{e_2},P_2,0)]}{\nu_2^{\boldsymbol{n}}+\lambda}\right. \\
&\qquad\left.+\frac{n_1r_1\tilde{V}(\boldsymbol{n}-\boldsymbol{e_1},P_2,0)+(n_2-1)r_2\tilde{V}(\boldsymbol{n}-\boldsymbol{e_2},P_2,0)}{\nu_2^{\boldsymbol{n}}+\lambda}\right] \\
&\quad-\left[\frac{R_1(\nu_1^{\boldsymbol{n}}+\lambda)+\mu_1[R_2+\tilde{V}(\boldsymbol{n}-\boldsymbol{e_1},P_2,0)]}{\nu_1^{\boldsymbol{n}}+\lambda}\right. \\
&\qquad\left.+\frac{(n_1-1)r_1\tilde{V}(\boldsymbol{n}-\boldsymbol{e_1},P_1,0)+n_2r_2\tilde{V}(\boldsymbol{n}-\boldsymbol{e_2},P_1,0)}{\nu_1^{\boldsymbol{n}}+\lambda}\right]
\end{aligned}
$$

The previous step uses condition (i), and the following step uses the definition of $\nu_j^{\boldsymbol{n}}$.

$$
\begin{aligned}
&=\left[\frac{R_2\lambda+\mu_2[R_2+R_2+\tilde{V}(\boldsymbol{n}-\boldsymbol{e_2},P_2,0)]}{\nu_2^{\boldsymbol{n}}+\lambda}\right. \\
&\qquad\left.+\frac{n_1r_1[R_2+\tilde{V}(\boldsymbol{n}-\boldsymbol{e_1},P_2,0)]+(n_2-1)r_2[R_2+\tilde{V}(\boldsymbol{n}-\boldsymbol{e_2},P_2,0)]}{\nu_2^{\boldsymbol{n}}+\lambda}\right] \\
&\quad-\left[\frac{R_1\lambda+\mu_1[R_1+R_2+\tilde{V}(\boldsymbol{n}-\boldsymbol{e_1},P_2,0)]}{\nu_1^{\boldsymbol{n}}+\lambda}\right. \\
&\qquad\left.+\frac{(n_1-1)r_1[R_1+\tilde{V}(\boldsymbol{n}-\boldsymbol{e_1},P_1,0)]+n_2r_2[R_1+\tilde{V}(\boldsymbol{n}-\boldsymbol{e_2},P_1,0)]}{\nu_1^{\boldsymbol{n}}+\lambda}\right]
\end{aligned}
$$

The following step again uses the assumptions in condition (i).

$$\geq \left[\frac{R_2\lambda + \mu_2[R_2 + R_1 + \tilde{V}(\boldsymbol{n} - \boldsymbol{e_2}, P_1, 0)]}{\nu_2^{\boldsymbol{n}} + \lambda} + \frac{n_1 r_1[R_2 + \tilde{V}(\boldsymbol{n} - \boldsymbol{e_1}, P_2, 0)] + (n_2 - 1)r_2[R_1 + \tilde{V}(\boldsymbol{n} - \boldsymbol{e_2}, P_1, 0)]}{\nu_2^{\boldsymbol{n}} + \lambda}\right]$$

$$- \left[\frac{R_1\lambda + \mu_1[R_1 + R_2 + \tilde{V}(\boldsymbol{n} - \boldsymbol{e_1}, P_2, 0)]}{\nu_1^{\boldsymbol{n}} + \lambda} + \frac{(n_1 - 1)r_1[R_2 + \tilde{V}(\boldsymbol{n} - \boldsymbol{e_1}, P_2, 0)] + n_2 r_2[R_1 + \tilde{V}(\boldsymbol{n} - \boldsymbol{e_2}, P_1, 0)]}{\nu_1^{\boldsymbol{n}} + \lambda}\right]$$

Rearranging these terms yields the following.

$$= \left[\frac{R_2(\mu_2 + \lambda)}{\nu_2^{\boldsymbol{n}} + \lambda} - \frac{R_1(\mu_1 + \lambda)}{\nu_1^{\boldsymbol{n}} + \lambda}\right] + \left[\left(R_1 + \tilde{V}(\boldsymbol{n} - \boldsymbol{e_2}, P_1, 0)\right)\left(\frac{\mu_2 + (n_2 - 1)r_2}{\nu_2^{\boldsymbol{n}} + \lambda} - \frac{n_2 r_2}{\nu_1^{\boldsymbol{n}} + \lambda}\right) + \left(R_2 + \tilde{V}(\boldsymbol{n} - \boldsymbol{e_1}, P_2, 0)\right)\left(\frac{n_1 r_1}{\nu_2^{\boldsymbol{n}} + \lambda} - \frac{\mu_1 + (n_1 - 1)r_1}{\nu_1^{\boldsymbol{n}} + \lambda}\right)\right]$$

The expression inside the first set of brackets is non-negative as a direct consequence of condition (ii), and leads to the proposed RTRI heuristic presented below. The remaining steps use the initial claim we made above, in addition to the requirement that λ be sufficiently small.

$$\geq \left(R_1 + \tilde{V}(\boldsymbol{n} - \boldsymbol{e_2}, P_1, 0)\right) * \left(\frac{\mu_1\mu_2 - r_1 r_2 + (n_1 - 1)r_1(\mu_2 - r_2) + (n_2 - 1)r_2(\mu_1 - r_1) + \lambda(\mu_2 - r_2)}{(\nu_2^{\boldsymbol{n}} + \lambda)(\nu_1^{\boldsymbol{n}} + \lambda)}\right)$$

$$- \left(R_2 + \tilde{V}(\boldsymbol{n} - \boldsymbol{e_1}, P_2, 0)\right) * \left(\frac{\mu_1\mu_2 - r_1 r_2 + (n_1 - 1)r_1(\mu_2 - r_2) + (n_2 - 1)r_2(\mu_1 - r_1) + \lambda(\mu_1 - r_1)}{(\nu_2^{\boldsymbol{n}} + \lambda)(\nu_1^{\boldsymbol{n}} + \lambda)}\right)$$

In this final expression, we know that $R_1 + \tilde{V}(\boldsymbol{n} - \boldsymbol{e_2}, P_1, 0) > R_2 + \tilde{V}(\boldsymbol{n} - \boldsymbol{e_1}, P_2, 0)$, with the difference growing as $\lambda \to 0$. However, the other two expressions

become equal as $\lambda \to 0$. Therefore we can argue that for some sufficiently small λ, the expression as a whole will be positive. ■

The inequality in condition (ii) defines a region similar to the one used in Argon et al.'s TRI rule. Before moving on to the heuristic motivated by condition (ii), it is important to discuss some implications and limitations of this result. First, we note that, unlike the corresponding condition in Argon et al., the numerator of the right hand side of condition (ii) can be negative for certain parameter values satisfying the initial assumptions. In these cases, the region in which class 2 is preferred becomes empty, and the heuristic always prioritizes class 1. Looking closely at the expression in question shows that this will happen when the parameters logically point toward choosing class 1, such as when R_1 is much larger than R_2, the expected lifetimes are similar, or λ becomes large.

The impact of λ on the lemma warrants further discussion. When λ is large (as $\lambda \to \infty$), the reward functions go to zero almost immediately, meaning that all future rewards will be negligible and priority should be given to the job with greater immediate reward (class 1). In this case, no heuristic should ever give preference to class 2, regardless of the expected lifetimes and service times. However, when $\lambda = 0$, the simplified version of Lemma 6.2 holds (the same proof can be used). This observation suggests that the lemma becomes applicable for some value of λ approaching 0 and necessitates the "for sufficiently small λ" condition in the lemma. Rather than compute bounds on λ for which the lemma applies (which would need to be in terms of the other input parameters), we will let negative values on the right hand side of condition (ii) indicate the instances in which the lemma (and the resulting heuristic) does not ever give priority to class 2.

We now return to the specifics of the heuristic suggested by condition (ii). Rearranging the inequality gives a rule that chooses the available job class j that minimizes the following expression:

$$\frac{1}{R_j}\left(1 + \frac{1}{\mu_j + \lambda} \sum_{i \in A_j(\boldsymbol{n})} (n_i - \delta_{ij}) r_i\right)$$

This expression is a weighted version of the original TRI rule that weights the approximate expected number of abandonments during service by the reward for service, with an additional adjustment for the rate at which the rewards are decaying. As a result, a job class that results in a large number of abandonments (due to a long service time) may still be selected if its reward is sufficiently high. By replacing R_j with a general reward function $R_j(t)$ and λ with the associated hazard rate function $\theta_j(t)$, we end up with a heuristic that can be applied to the Markov case with general reward functions. The resulting "RTRI" (triangular with rewards) rule chooses the available job class j that minimizes the following expression:

$$\frac{1}{R_j(t)}\left(1+\frac{1}{\mu_j+\theta_j(t)}\sum_{i\in A_j(\boldsymbol{n})}(n_i-\delta_{ij})r_i\right)$$

Minimum Losses During Service Rule

As mentioned in the previous subsection, the triangular region defined by the TRI (RTRI) rule effectively minimizes the approximate expected (weighted) number of abandonments during service. This observation suggests a heuristic that more directly focuses on the losses during service, rather than arrive at an approximation through structural results. Unlike the TRI and RTRI rules which must make strong assumptions about the reward functions in order to perform structural analyses, the following proposed Minimum Losses During Service (MLDS) rule begins with no additional assumptions about the reward functions, other than the initial requirement that they be positive and non-increasing. When a job of class j is selected for service beginning at time t, the server earns the reward $R_j(t)$. In the course of this job's service, the server loses the full reward $R_i(t)$ for each job of class i whose lifetime expires during service. At the same time, the server loses a portion of the reward ($R_i(t)-R_i(t+s)$, if s is the duration of service) for each job in the queue whose lifetime does not expire during service. The total expected reward lost by selecting a job from class j can be expressed as follows.

$$-R_j(t) + \sum_{i \in A_j(\boldsymbol{n})} (n_i - \delta_{ij}) \int_0^\infty \Big[R_i(t) G_j(s) + \big(R_i(t) - R_i(t+s)\big)\big(1 - G_j(s)\big)\Big] dF_j(s)$$

$$= -R_j(t) + \sum_{i \in A_j(\boldsymbol{n})} (n_i - \delta_{ij}) \int_0^\infty \Big[R_i(t) - \big(1 - G_j(s)\big) R_i(t+s)\Big] dF_j(s)$$

$$= -R_j(t) + \sum_{i \in A_j(\boldsymbol{n})} (n_i - \delta_{ij}) \left[R_i(t) - \int_0^\infty \mu_j e^{-(r_i + \mu_j)s} R_i(t+s) ds \right]$$

It is informative to look at what this expression simplifies to under some of the assumptions we have considered on the reward functions (or will consider during testing). If the rewards diminish exponentially at different rates to different non-zero values (i.e. $R_j(t) = b_j + (a_j - b_j)e^{-\lambda_j t}$ for initial reward a_j and asymptotic reward b_j), then the MLDS rule simplifies to choosing the job class j that minimizes the following:

$$-R_j(t) + \sum_{i \in A_j(\boldsymbol{n})} (n_i - \delta_{ij}) \left[b_i \left(\frac{r_i}{r_i + \mu_j} \right) + (a_i - b_i) e^{-\lambda_j t} \left(\frac{\lambda_i + r_i}{\lambda_i + r_i + \mu_j} \right) \right]$$

If the rewards decay exponentially to zero at the same rate ($b_j = 0$ and $\lambda_j = \lambda \ \forall j$), then MLDS chooses the job class j that minimizes the following:

$$-R_j(t) + \sum_{i \in A_j(\boldsymbol{n})} (n_i - \delta_{ij}) R_i(t) \left(\frac{\lambda + r_i}{\lambda + r_i + \mu_j} \right)$$

From this expression, we can easily see what the rule would become if rewards are constant ($R_j(t) = R_j$ and $\lambda_j = 0 \ \forall j$) or uniform ($R_j(t) = 1 \ \forall j$). Here it becomes easier to see the differences between this rule and the RTRI rule discussed above. The abandonments from each job class are weighted by the reward associated with that job class, rather than all classes being weighted by the reward of the job in service. Furthermore, the rates at which the rewards for each class are diminishing appear directly in the calculation. Finally, if the rewards are uniform and constant (as in the

problem studied by Argon et al.), this rule reduces to the exact expected number of abandonments during service, rather than an approximation.

Fluid-Based Policy Improvement Heuristic

Li and Glazebrook (2010) also study the problem with constant, uniform rewards ($R_j(t) = 1 \ \forall j$) and propose an approximate dynamic programming heuristic to improve on the rules studied by Glazebrook et al. (2004) and Argon et al. (2008). Their work combines two commonly used approaches: (1) using a single policy improvement step to improve some given heuristic (see, for example, Tijms 1994) and (2) approximating the dynamic programming value function associated with a given policy (see, for example, Powell 2007).

The single-step policy improvement concept works within the framework of the SDP optimality equation in the following manner, using the notation from formulation (1) above and incorporating the diminishing reward functions. The value associated with following a scheduling policy π beginning at decision state $(\boldsymbol{n}, t)$ (prior to the decision being made) can be defined as $V_\pi(\boldsymbol{n}, t)$. Let $\pi(\boldsymbol{n}, t)$ be the decision made by policy π at decision state $(\boldsymbol{n}, t)$. π can potentially be improved by a policy improvement step that allows deviations from π for the current decision but assumes that π will continue to be used at all future decision epochs. The resulting policy improvement (PI) decision for policy π at state $(\boldsymbol{n}, t)$ can be defined as follows.

$$\pi_{PI}(\boldsymbol{n}, t) = \underset{j \mid n_j > 0}{\arg\max} \left\{ R_j(t) + \int_0^\infty \sum_{\boldsymbol{n}'} p(\boldsymbol{n}' | \boldsymbol{n}, t, j, s) \, V_\pi(\boldsymbol{n}', t + s) \, dF_j(s) \right\}$$

Notice that the value function used at all future system states is based on using policy π, rather than the optimal policy. However, as Li and Glazebrook acknowledge, there is considerable difficulty involved in computing $V_\pi(\boldsymbol{n}, t)$ for all possible future states, even if π is a static policy (they use the $r\mu$ rule in their study). The novelty of their approach lies in the use of a fluid approximation $V_\pi^F(\boldsymbol{n}, t)$ for the value function $V_\pi(\boldsymbol{n}, t)$, which results in a single-step fluid-based, policy improvement heuristic

(FPI). Using this fluid approximation, the resulting FPI decision for policy π in state $(\boldsymbol{n}, t)$ is given by:

$$\pi_{FPI}(\boldsymbol{n}, t) = \underset{j \mid n_j > 0}{\arg\max} \left\{ R_j(t) + \int_0^{\infty} \sum_{\boldsymbol{n}'} p(\boldsymbol{n}' | \boldsymbol{n}, t, j, s)\, V_\pi^F(\boldsymbol{n}', t+s)\, dF_j(s) \right\}$$

This rule represents the logical extension of the heuristic proposed by Li and Glazebrook to the problem with diminishing rewards. It is clear that the first challenge involved in this extension is the need to incorporate the reward functions into the fluid approximation, $V_\pi^F(\boldsymbol{n}, t)$. Diminishing rewards can make even the $Rr\mu$ rule into a dynamic rule, and Li and Glazebrook define their fluid approximation based on static rules. Our modification of their fluid algorithm will allow for dynamic rules within the approximation. We also observe that this version of the FPI rule is still computationally intensive for general reward functions. Even in the Markov case, general reward functions force the integration over the service time to be done numerically, which results in repeated calls to the fluid approximation (which must be computed "online" due to the dependence of the rewards on the time dimension).

To address this issue, we propose a further modification of the FPI heuristic based on the structure of formulation (2) that greatly reduces its computational burden. The FPI rule above explicitly enumerates all possible sources of stochasticity due to the selection of a job class (both the length of service and the number of abandonments during service), and this enumeration is the chief cause of the computational difficulty. One way to avoid this difficulty is to incorporate the expected service time of the selected job into the fluid approximation. In the notation of formulation (2), the result is a fluid approximation of $\tilde{V}_\pi(\boldsymbol{n}, P_j, t)$ (with a job from class j already selected for service) rather than an approximation of $V_\pi(\boldsymbol{n}, t)$. Let $\tilde{V}_\pi^F(\boldsymbol{n}, P_j, t)$ be this new fluid approximation function. The resulting, modified FPI decision for policy π in state $(\boldsymbol{n}, t)$, which we will be the one we use in our testing in Section 6.5, becomes:

$$\tilde{\pi}_{FPI}(\boldsymbol{n}, t) = \underset{j \mid n_j > 0}{\arg\max} \left\{ R_j(t) + \tilde{V}_\pi^F(\boldsymbol{n}, P_j, t) \right\}$$

The similarity with the optimality equations in formulation (2) is clear, as are the computational savings from only computing one approximation for each job class rather than one for each possible (discretized) service time from each possible job class. Initial computational testing illustrates the computational savings involved and shows no significant drop-off in solution quality (refer to Table A-24 and Table A-25 in the appendix). It is important to note that FPI must be paired with an initial rule (π) in order to generate a fully-functioning heuristic. Our computational testing will pair FPI with several of the heuristics proposed in the sections above.

We now proceed with the details of the modified fluid approximation itself, which allows for a dynamic underlying policy π and incorporates the service time of a job already selected for service. The fluid approximation deterministically models all service times according to their means and treats the number of each job class present in the queue as a continuous variable that decreases (i.e. abandons the queue) at a deterministic rate according to the parameters of its lifetime distribution. At the completion of every service, the underlying policy π is used to select the next job to be served. If a job of class j is selected for service in state $\boldsymbol{n}$, and we continue to assume that all random variables are exponentially distributed, then the amount of job class i remaining in the system at the end of service (which lasts μ_j^{-1}) is given by $(n_i - \delta_{ij})e^{-r_i\mu_j^{-1}}$. The algorithm continues selecting jobs for service until the system is empty. To account for the fact that the continuous version of n_i can be less than one, we allow fractional service times and rewards and allow the underlying policy π to determine priorities based on fractional states. The full algorithm for computing the fluid approximation for the value of being in state $(\boldsymbol{n}, Q, t)$ is given below.

Fluid Approximation Algorithm for $\tilde{V}_\pi^F(\boldsymbol{n}, Q, t)$:

INITIALIZE $\tilde{V}_\pi^F(\boldsymbol{n}, Q, t) = 0$ // Initialize value to zero

IF $Q = P_j$ // If job already in service

$n_i = (n_i - \delta_{ij})e^{-r_i\mu_j^{-1}}$ // Update jobs left in queue

$t = t + \mu_j^{-1}$ // Advance time

END-IF

WHILE $\sum_{i=1}^{K} n_i > 0$ // While system not empty

Let $j = \pi(\boldsymbol{n}, t)$ // Choose job for service

$\Delta_j = \min(n_j, 1)$ // Account for fractional remains

$\tilde{V}_\pi^F(\boldsymbol{n}, Q, t) = \tilde{V}_\pi^F(\boldsymbol{n}, Q, t) + \Delta_j R_j(t)$ // Reap reward

$n_i = (n_i - \Delta_j\delta_{ij})e^{-r_i\mu_j^{-1}}$ // Update jobs left in queue

$t = t + \Delta_j\mu_j^{-1}$ // Advance time

END-WHILE

RETURN $\tilde{V}_\pi^F(\boldsymbol{n}, Q, t)$ // Return approximate value function

Note that this algorithm is written for the general state $(\boldsymbol{n}, Q, t)$, allowing the same algorithm to be used for $\tilde{V}_\pi^F(\boldsymbol{n}, P_j, t)$ (in the modified FPI) and for $\tilde{V}_\pi^F(\boldsymbol{n}, E, t) = V_\pi^F(\boldsymbol{n}, t)$ (in the original FPI).

Generalizing to the non-Markov Case

While we have taken care to point out how each of the proposed heuristics can be applied with general reward functions, it is also important to discuss how they can be applied to problems with general lifetime and service time distributions. Extending the $R(r + \lambda)\mu$ and RTRI rules can be accomplished by replacing the service and lifetime rates (μ_j and r_j, respectively) with the reciprocals of the expected service times and remaining lifetimes. The expected remaining lifetimes can be computed conditionally at each decision epoch t. With these changes, at decision epoch t, the $R(r + \lambda)\mu$ selects the available job class j that maximizes the following.

$$R_j(t)\left(\frac{1}{E\left[Y_j - t | Y_j > t\right]} + \theta_j(t)\right)\frac{1}{E\left[X_j\right]}$$

Using these same replacements for the service and lifetime rates, at decision epoch t, the RTRI heuristic chooses the job class j that minimizes the following.

$$\frac{1}{R_j(t)}\left(1 + \frac{1}{\frac{1}{E\left[X_j\right]} + \theta_j(t)} \sum_{i \in A_j(\boldsymbol{n})} \frac{\left(n_i - \delta_{ij}\right)}{E\left[Y_j - t | Y_j > t\right]}\right)$$

The development of the MLDS heuristic in section 6.3 begins by assuming general lifetime and service time distributions. Therefore, the most general form of the rule prioritizes the available job class that minimizes the following.

$$-R_j(t) + \sum_{i \in A_j(\boldsymbol{n})} \left(n_i - \delta_{ij}\right) \int_0^\infty \left[R_i(t) - \left(1 - G_j(s)\right) R_i(t+s)\right] dF_j(s)$$

This generalization keeps the spirit of the original rule by explicitly minimizing the losses during service. However, in the event that this integration must be done numerically and is too computationally costly, the rule can also be generalized by replacing the service and lifetime rates from the Markov version with the reciprocals of the expected service times and remaining lifetimes.

Finally, we turn to the algorithm for the fluid approximation. Because service times are treated as deterministic in the fluid model, the service time component of the algorithm requires no modification. In fact, the only part requiring generalization is the computation of the number of jobs of each class remaining at the end of a given service. As discussed in Li and Glazebrook (2010), we observe that for service of class j starting with n_i jobs from class i in the system at time t, the number of jobs of class i remaining at time $s \geq t$ is the solution to the initial value problem

$$n_i(t) = n_i - \delta_{ij}$$
$$n_i'(s) = -r_i(s) n_i(s)$$

where $r_i(s)$ is the hazard rate function for the lifetime distribution for job class i (as discussed for the reward functions above). The solution to this initial value problem is given by

$$n_i(\tau) = (n_i - \delta_{ij})e^{-\int_t^{t+\tau} r_i(s)ds}$$

which can be substituted directly into algorithm given above.

6.4. Generating MCI Triage Instances

In order to test the performance of the heuristics proposed in the preceding section, we generate a set of realistic problem instances from the literature on MCI response planning and triage. Traditional triage algorithms separate survivors into four treatment categories (Frykberg 2005, Lerner et al. 2008), and each of these categories can be viewed as a class of patients with an associated lifetime (while waiting for treatment), treatment time, and deteriorating survival probability (depending on when treatment is received). Table 16 provides a qualitative assessment of how these class profiles might change across triage categories (excluding the *dead* category) based on their descriptions in the literature. In the day-to-day operation of a trauma center, where resources are more readily available, patient prioritization is primarily aligned with the patient's expected lifetime. During a MCI, where resources are stretched thin and the most important distinction is between those who should be treated immediately and those who can be delayed (Frykberg 2005), prioritization must be based on the full patient profiles of all the casualties. The *expectant* category, which is not typically used during day-to-day operations, acknowledges this reality by not prioritizing MCI casualties that will consume large amounts of resources (represented by their treatment times) with a low likelihood of survival. Thus, our analysis does not consider the expectant category. Likewise, we do not consider the walking wounded category because they can wait almost indefinitely with minimal loss of reward. The key decision considered in this paper is the choice between the immediate and the delayed categories.

Table 16. Lifetime, treatment time, and deteriorating survival probability profiles for traditional triage categories

Triage Category	Lifetime	Treatment Time	Survival Probability	
			Initial	Deterioration
Walking Wounded	very long	very short	very high	very slow
Delayed	long	short	high	slow
Immediate	short	long	moderate	fast
Expectant	very short	very long	low	very fast

In keeping with the spirit of the need for immediate vs. non-immediate prioritization, we begin our analysis of the MCI triage clearing system on problems involving two classes of patients. We assume that patients in class 2 are more severely wounded than patients in class 1. In other words, patients in class 2 will have shorter expected lifetimes, longer expected treatment times, and lower long-term survival probabilities. Figure 12 illustrates how such qualitative assessments (similar to those in Table 16) might translate to survival probabilities while waiting for treatment ($P(Y_j > t)$ for lifetime random variable Y_j) and after treatment ($R_j(t)$) for two job classes. Based on these survival probabilities alone, it might appear that class 2 should be prioritized over class 1 (at least at time 0). However, if class 2's service time is excessively long or if there are a large number of class 1 patients requiring treatment, prioritizing class 2 may be the wrong choice. As this example demonstrates, making triage decisions in order to maximize the expected number of survivors requires a clearer understanding of how total casualty loads and their associated lifetimes, treatment times, and long-term survival probabilities combine to dictate prioritization.

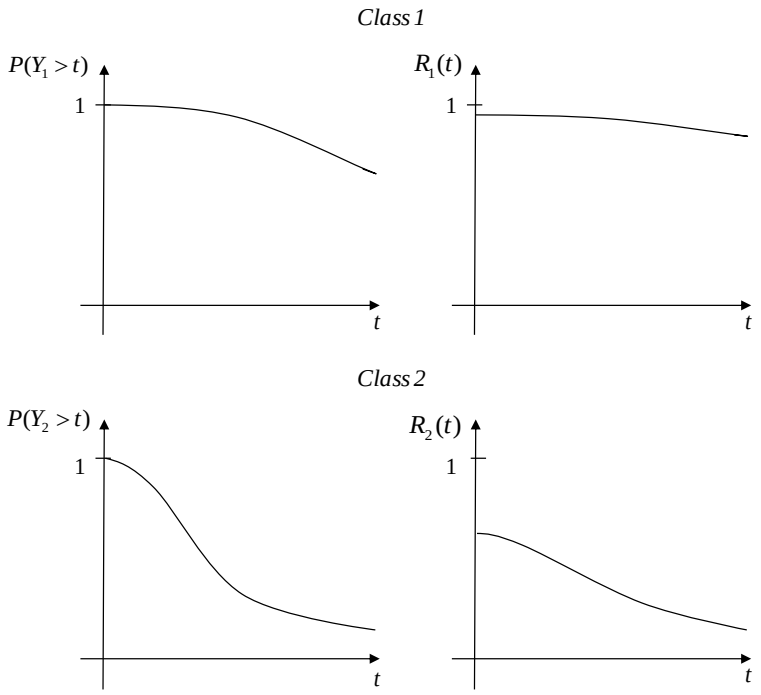

Figure 12. Potential pre-treatment and post-treatment survival probabilities for two patient classes

The literature on MCI response planning contains data that we can use to generate a wide range of scenarios for the MCI triage clearing system with two job classes. While higher profile MCIs are often associated with total casualties (killed and injured) in the hundreds, the literature suggests that most events are associated with much smaller casualty loads. For instance, studies of the two waves of terrorist bombings in Israel in the 1990's and early 2000's indicate that hospitals are typically faced with around 30 casualties and rarely face more than 50 (Ashkenazi et al. 2008, Kosashvili et al. 2009). The proportion of MCI casualties that are severely wounded (reflecting a need for immediate care) ranges from approximately 10 to 50 percent (Frykberg 2002, Hupert et al. 2007). The treatment times used in models to study MCI response plans indicate that severely injured patients (class 2) can require treatment lasting from half an hour to over two hours, while non-severely injured patients (class 1) require from five minutes to half an hour (Hirshberg et al. 1999, Hupert et al. 2007). For our analysis, these estimates must be adjusted down to reflect the fact that the treatment times in our single-server clearing system are essentially modeling the departure process from a much larger, more complex trauma system.

The literature contains very little data on how long patients are capable of surviving while waiting for treatment, so we consider a wide range of what we consider to be reasonable lifetimes (one to two hours for severely wounded patients, four to twelve hours for non-severely wounded patients). The range of scenarios we will consider for the number of jobs, expected service (treatment) times, and expected lifetimes for our scenarios are shown in Table 17.

Table 17. Parameters combinations for 4,320 test instances with two job classes

Parameter	# of Scenarios	Values
Number of Jobs, (n_1, n_2)	4	(20, 10); (20, 15); (30, 15); (30, 20)
Mean Lifetimes, (r_1^{-1}, r_2^{-1})	8	{240, 480, 720, 960} x {60, 120}
Mean Service Times, (μ_1^{-1}, μ_2^{-1})	9	{5, 10, 15} x {20, 25, 30}
Rewards, $R_j(t) = b_j+(a_j-b_j)\, e^{-\lambda_j t}$		
(a_1, b_1)	3	(0.9, 0.7); (0.98, 0.9); (0.98, 0.7)
(a_2, b_2)	5	(0.5, 0.4); (0.5, 0.2); (0.8, 0.6); (0.8, 0.4); (0.8, 0.2)
$(\lambda_1^{-1}, \lambda_2^{-1})$	1	(180, 60)

Estimates of long-term survival probabilities and their deterioration over time are derived from those used in the Sacco Triage Method (Sacco et al. 2005). Their research uses logistic regression to estimate the long-term survival probabilities associated with different RPM scores (a measure of **r**espiratory rate, **p**ulse rate, and **m**otor response), and then applies the Delphi technique (a method for achieving consensus among experts) to estimate how RPM scores deteriorate over time. A subset of the resulting survival probability curves (associated with different initial RPM scores) is shown in Figure 13, illustrating how long-term survival probabilities decrease as a function of the time treatment is initiated. Our analysis models diminishing rewards (survival probabilities) as exponential decay functions of the form $R_j(t) = b_j + (a_j - b_j)e^{-\lambda_j t}$. This form was chosen because it has few parameters (three for each job class) and is not computationally intensive when used in our proposed heuristics. The survival probability scenarios we will use for the

diminishing reward functions in our analysis are also shown in Figure 13, and the parameters for these functions are included in Table 17.

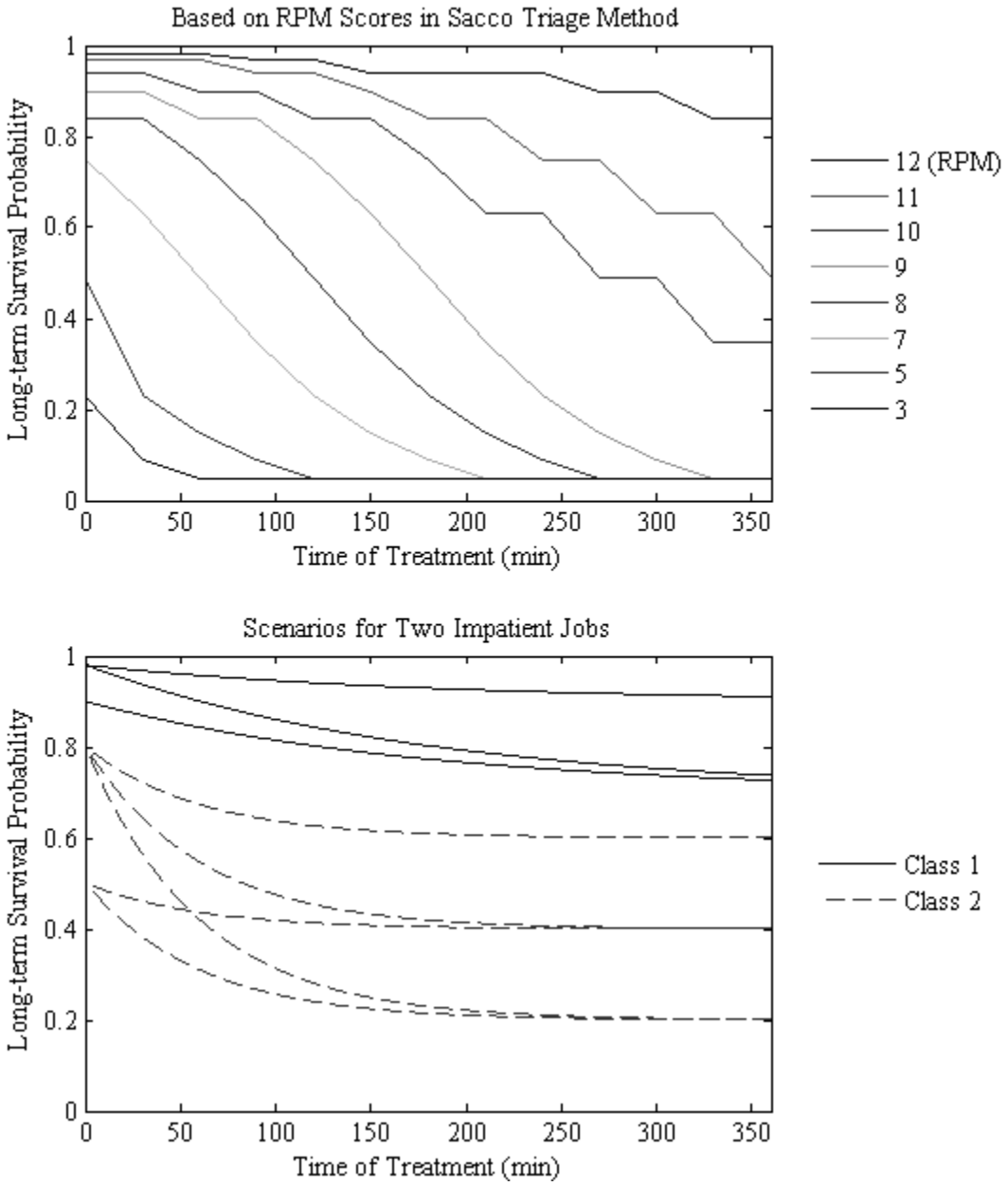

Figure 13. Deterioration of long-term survival probabilities used in the Sacco Triage Method and the proposed reward functions for the MCI triage clearing system

6.5. Computational Testing

The range of parameters presented in Table 17 can be used to create test problems that allow us to analyze how triage prioritization decisions depend on patient information (survival while waiting for treatment, time required for treatment, and long-term survival probability) and on the number of patients seeking treatment. Because problems with exponential random variables and rewards that decay exponentially to zero at a uniform rate (as discussed in Section 6.2) can be solved to optimality, we begin by looking at three sets of such problems with increasingly large rates of decay. For these problems, we compare the performance of each of the non-fluid-based heuristics with the performance of the optimal policy. We then proceed with a full analysis of the heuristics, including the fluid-based approaches, across the more general set of problems defined by Table 17. In each case, a discrete-event simulation of the system is used to compare the performance of the proposed rules with each other and, when available, with the optimal solution.

Uniformly Decaying Rewards

Combining each of the problem sizes, mean service times, mean lifetimes, and initial reward values ($R_j(0) = a_j$) in Table 17 yields a set of 1,152 different problem instances for the case in which all rewards decay to zero at the same rate. We combine this set of parameters with three different rates of decay ($\lambda = 0$ and $\lambda^{-1} = 180, 60$) to generate a larger set of 3,456 test instances. Note that $\lambda = 0$ is equivalent to the problem with constant rewards, as studied by Glazebrook et al. (2004). For each of these problems we tested the non-fluid-based heuristics presented in Section 6.3 by computing the optimal policy and then simulating the performance of the optimal policy and the heuristics. For each of the rules, we tested the version without rewards ($r\mu$, TRI, and MLDS) and the version(s) with rewards incorporated ($Rr\mu$, $R(r + \lambda)\mu$, RTRI, RMLDS) in order to explore the value of incorporating reward information into prioritization decisions. Although not discussed above, we also tested the performance of the Shortest Expected Processing Time (SEPT) rule, which is commonly used in the job scheduling literature as a baseline heuristic. SEPT will always give priority to the class 1 jobs. For each problem instance, the performances

of each of the eight resulting rules are averaged over 5,000 simulation replications and compared with the mean performance of the optimal policy in the form of percentage deviations. The computational effort for these non-fluid-based rules is minimal, with each rule averaging roughly 0.0003 seconds per replication.

In the discussion that follows, we will make use of policy maps to gain insight into the performance of certain rules. Sample policy maps for the optimal solution and for the TRI rule for one of the test problems with constant rewards are shown in Figure 14. The policy maps show which class of job each policy will choose in all possible system states. The four different initial problem sizes (N_1, N_2) are indicated in each map. For the selected problem, we see that the optimal policy prioritizes class 2 in most states, while the TRI rule only prioritizes class 2 in a small number of states. The mean percentage deviations from the optimal policy for the TRI rule for initial sizes $(N_1, N_2) = (20,15)$ and $(N_1, N_2) = (30,20)$ are 5.54 and 2.97, respectively.

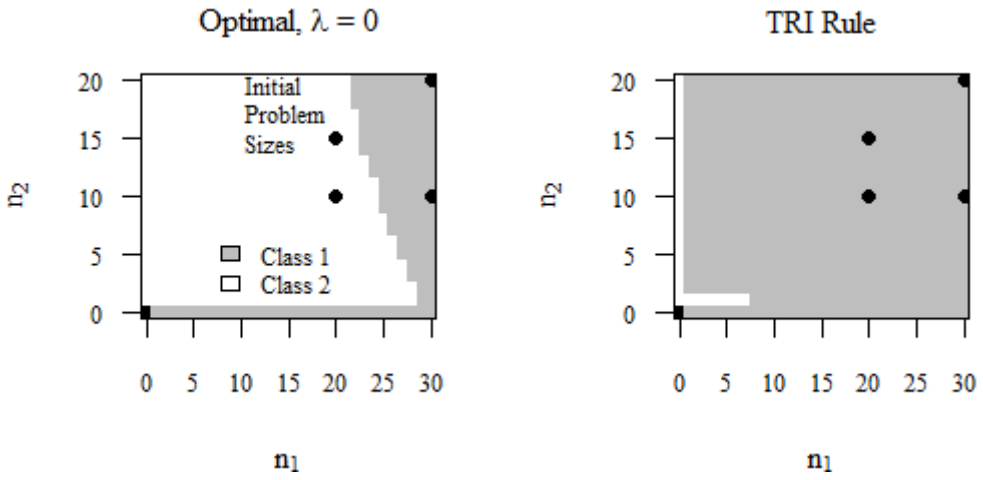

Figure 14. Policy maps for optimal solution and TRI rule for a problem with uniformly decaying rewards, $(\mu_1^{-1}, \mu_2^{-1}) = (10,20)$, $(r_1^{-1}, r_2^{-1}) = (480,60)$, $(R_1, R_2) = (0.9,0.8)$, and $\lambda = 0$

Table 18 shows the mean and maximum percentage deviations and the number of times that the rule solves the problem optimally for each of the heuristics over the full set of 1,152 problems for each value of λ. Both the mean deviations and worst-case performances for $r\mu$ and $Rr\mu$ increase dramatically for larger values of λ. Of the $r\mu$-

based rules, only $R(r+\lambda)\mu$ does not dramatically deteriorate as λ increases. The mean deviation for the $R(r+\lambda)\mu$ rule for $\lambda^{-1}=60$ is less than one percent and the heuristic finds the optimal solution in 97 percent of cases. In spite of this, the maximum deviation remains high, suggesting that the $r\mu$-based rules are, in some sense, "all or nothing" heuristics. It is also clear that the $r\mu$ rule gets progressively better as more information about the reward function is incorporated (first R, then λ).

The performance of SEPT for the problems in Table 18 is remarkably similar to the performance of the TRI, RTRI, MLDS, and RMLDS rules, and in fact, the correlation coefficients between the percentage deviations for these five rules for all λ's exceed 0.98 (coefficients not shown). In contrast with the $r\mu$-based heuristics, these rules all improve as λ increases, with SEPT, RTRI, and RMLDS finding optimal solutions in 98 percent of the problems when $\lambda^{-1}=60$. It is interesting to note that for the problems with constant rewards, adding the reward information to the TRI and MLDS heuristics actually makes their performance worse. However, when $\lambda>0$, both RTRI and RMLDS do better than TRI and MLDS, respectively. This difference is chiefly due to the fact that RTRI and RMLDS exactly match SEPT for all problems with $\lambda>0$, and SEPT becomes increasingly optimal as λ increases.

Table 18. Summary statistics for percentage deviations from optimal policy for non-fluid-based heuristics applied to problems with uniformly decaying rewards

	SEPT	$r\mu$	$Rr\mu$	$R(r+\lambda)\mu$	TRI	RTRI	MLDS	RMLDS
$\lambda = 0$								
Mean	1.53	4.94	2.62	2.62	1.22	1.53	1.21	1.52
Maximum	15.09	29.22	20.63	20.63	12.38	15.09	12.29	14.43
% Optimal*	31.8	27.2	43.1	43.1	15.3	31.8	17.6	31.8
λ^{-1} =180								
Mean	0.01	30.41	23.22	3.83	0.30	0.01	0.26	0.01
Maximum	0.65	64.22	57.33	35.83	4.04	0.65	4.73	0.65
% Optimal*	84.2	12.5	28.5	81.6	13.1	84.2	15.7	84.2
λ^{-1} = 60								
Mean	0.00	48.44	37.94	0.72	0.30	0.00	0.27	0.00
Maximum	0.01	84.36	79.34	35.40	5.62	0.01	6.70	0.01
% Optimal*	98.0	12.5	28.5	97.2	12.6	98.0	15.4	98.0

* Percent of the problems for which each rule generates the optimal solution.

In order to shed some light on how the SEPT, TRI and RTRI rules relate to the optimal policies, we turn to a set of policy maps for one of the problems that SEPT

does not solve optimally for any of the tested values of λ. The maps in Figure 15 clearly show the effect that increasing λ has on the optimal decisions, moving the optimal policy (for the states shown) from always prioritizing class 2 to almost always prioritizing class 1. This trend explains why SEPT is nearly always optimal when $\lambda^{-1} = 60$. As in Figure 14, the maps in Figure 15 show that when $\lambda = 0$, the TRI rule (which does not use any reward information) significantly underestimates the region in which class 2 should be prioritized. Incorporating rewards with $R_1 > R_2$ into the RTRI rule further reduces the size of the region where class 2 is selected, thus pushing the rule further from the optimal solution. However, the optimal policies begin to look more like SEPT when $\lambda > 0$, so reducing the size of the region where class 2 is selected becomes a good idea for these instances.

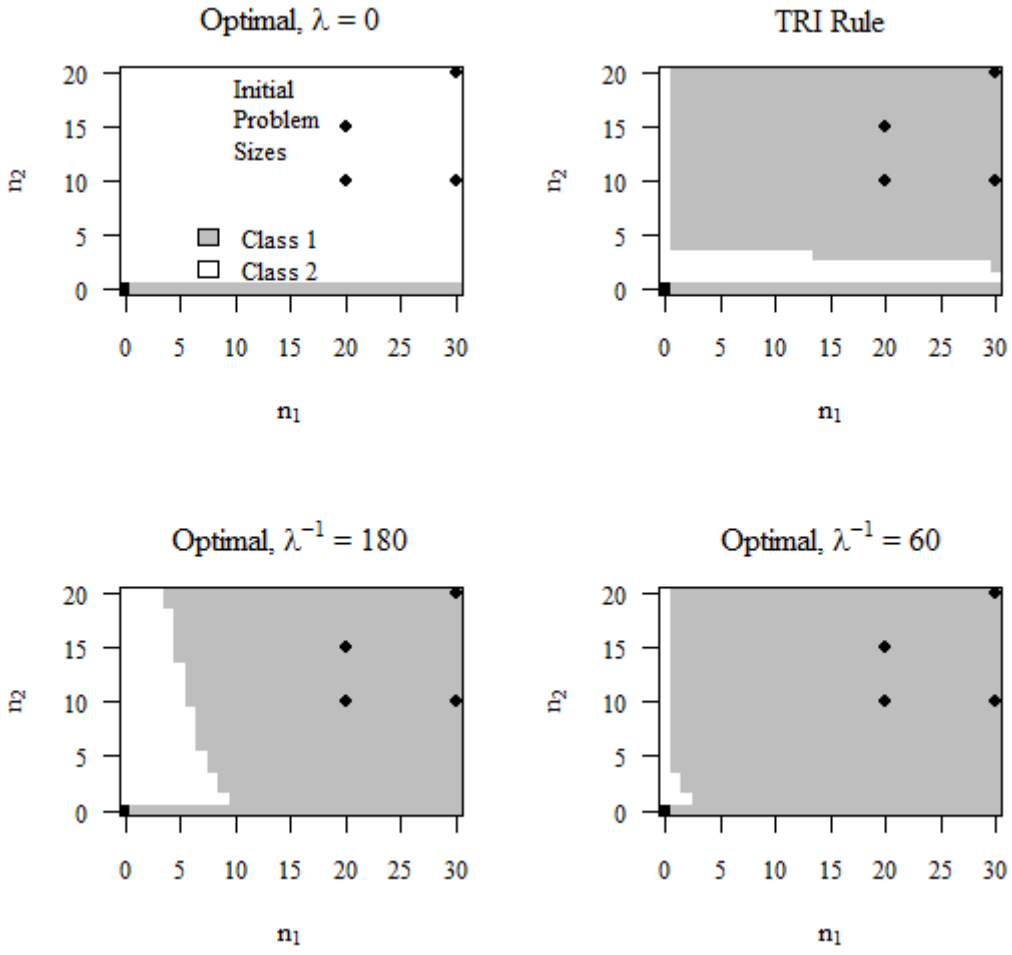

Figure 15. Policy maps for optimal solutions and TRI rule for test problems with $(\mu_1^{-1}, \mu_2^{-1}) = (15,20)$, $(r_1^{-1}, r_2^{-1}) = (960,60)$, $(R_1, R_2) = (0.9,0.8)$, $\lambda = 0, \frac{1}{180}$, and $\frac{1}{60}$

These observations about the impact of adding rewards to the TRI heuristic prompt a comment on the limitations of Lemma 6.2, at least for the range of problem instances we are studying. RTRI matches SEPT exactly for each of the problems in this initial analysis, indicating that the region defined by Lemma 6.2, condition (ii) is empty for problems with parameters pulled from the MCI response planning and triage literature. This observation does not contradict the claims of the lemma, but it does suggest that for MCI triage-motivated problems, no feasible state (n_1, n_2) satisfies the inequality in condition (ii). It is clear from Lemma 6.2 that for some set of parameters the RTRI rule will not reduce to SEPT, but searching for these parameters is beyond the scope of this research.

The means presented in Table 18 are taken over 1,152 test problems constructed from of a wide range of problem sizes, expected service times, expected lifetimes, and initial reward values. The results in Table 19 show how the performances of the different rules vary over the different values for these parameters for the problems with $\lambda = 0$. For tables showing similar analyses for the problems with $\lambda > 0$, which show comparable, albeit muted, trends, please see Table A-26 and Table A-27 in the appendix. The performance of all the heuristics gets worse as the number of class 2 jobs increases with respect to a fixed number of class 1 jobs. The $r\mu$-based rules similarly get worse as the total number of jobs increases, while the other five rules all get better as the total number of jobs increases. The trends across the other parameters continue to be divided between the $r\mu$-based rules and the SEPT, TRI, and MLDS rules. The $r\mu$-based rules all do better when the difference between the expected lifetimes gets larger and when the differences between the expected service times and between the reward values get smaller. This is in line with the results from Glazebrook et al. (2004), who show that $Rr\mu$ is asymptotically optimal when the expected lifetimes go to infinity. In contrast, SEPT, TRI, RTRI, MLDS, and RMLDS all improve their performance when the expected lifetimes become smaller and more similar and when the differences between the expected service times and between the reward values become larger. This is line with the results of Argon et al. (2008), who find that the TRI rule works best when jobs are time-critical (i.e., when jobs have shorter expected lifetimes).

Table 19. Mean percentage deviations from optimal policy for non-fluid-based heuristics for problems with $\lambda = 0$ by different input parameters

	SEPT	$r\mu$	$Rr\mu$	$R(r+\lambda)\mu$	TRI	RTRI	MLDS	RMLDS
(n_1, n_2)								
(20,10)	1.81	3.95	1.95	1.95	1.42	1.81	1.40	1.79
(20,15)	2.02	4.86	2.46	2.46	1.66	2.02	1.64	2.00
(30,10)	0.98	4.62	2.55	2.55	0.75	0.98	0.77	0.97
(30,20)	1.30	6.33	3.52	3.52	1.04	1.30	1.05	1.29
(r_1^{-1}, r_2^{-1})								
(240,60)	0.29	9.88	5.58	5.58	0.20	0.29	0.19	0.29
(480,60)	1.29	5.80	4.31	4.31	0.90	1.29	0.91	1.27
(720,60)	2.44	2.48	2.48	2.48	1.82	2.44	1.82	2.41
(960,60)	3.53	1.12	1.12	1.12	2.71	3.53	2.69	3.47
(240,120)	0.02	6.74	0.45	0.45	0.03	0.02	0.03	0.02
(480,120)	0.59	5.93	3.02	3.02	0.51	0.59	0.51	0.59
(720,120)	1.54	4.93	2.31	2.31	1.34	1.54	1.34	1.54
(960,120)	2.52	2.65	1.69	1.69	2.23	2.52	2.23	2.52
(μ_1^{-1}, μ_2^{-1})								
(5,20)	0.45	6.23	1.91	1.91	0.42	0.45	0.43	0.45
(5,25)	0.16	3.95	1.78	1.78	0.15	0.16	0.15	0.16
(5,30)	0.06	5.37	2.36	2.36	0.06	0.06	0.06	0.06
(10,20)	2.43	4.63	2.57	2.57	2.15	2.43	2.16	2.43
(10,25)	1.27	4.56	2.62	2.62	1.09	1.27	1.10	1.27
(10,30)	0.68	6.60	4.22	4.22	0.57	0.68	0.58	0.68
(15,20)	4.50	2.22	1.32	1.32	3.18	4.50	3.13	4.43
(15,25)	2.65	4.32	2.72	2.72	2.11	2.65	2.08	2.62
(15,30)	1.55	6.59	4.08	4.08	1.23	1.55	1.25	1.53
(R_1, R_2)								
(0.98,0.8)	2.25	3.35	2.17	2.17	1.83	2.25	1.83	2.25
(0.98,0.5)	0.39	7.19	3.45	3.45	0.24	0.39	0.23	0.39
(0.9,0.8)	2.91	2.72	1.91	1.91	2.43	2.91	2.44	2.86
(0.9,0.5)	0.56	6.49	2.96	2.96	0.37	0.56	0.36	0.56

Because of the large number of problems for which the optimal policy follows SEPT, we return to the mean deviations presented in Table 18 and explore how they vary with the optimality of SEPT. Table 20 divides the instances for each value of λ according to the optimality of SEPT and shows the mean percentage deviations for each of the rules within each subset. In addition, for problems that SEPT does not solve optimally, the table presents the percentages of problems that each rule solves optimally and for which each rule finds the best heuristic solution. Among these

problems, the TRI, RTRI, MLDS, and RMLDS rules continue to outperform the $r\mu$-based rules, although for $\lambda = 0$ the $r\mu$-based rules are much more likely to find the optimal and best heuristic solutions. For the problems with $\lambda^{-1} = 60$, we see that SEPT always finds the best heuristic solution even when it is not optimal.

Table 20. Mean percentage deviations from the optimal policy for non-fluid-based heuristics by optimality of SEPT, and percentages of problems for which SEPT is suboptimal that the rules find the optimal and best heuristic solution

	N	SEPT	$r\mu$	$Rr\mu$	$R(r+\lambda)\mu$	TRI	RTRI	MLDS	RMLDS
$\lambda = 0$, Means									
SEPT Optimal	366		8.48	1.18	1.18	0.01	0	0.01	0
SEPT Suboptimal	786	2.24	3.29	3.29	3.29	1.78	2.24	1.78	2.22
% Optimal*			21.5	21.5	21.5	1.7	0	1.5	0
% Best*		5.3	39.3	39.3	39.3	44.7	5.3	33.3	5.3
λ^{-1} =180, Means									
SEPT Optimal	970		32.40	23.86	0.84	0.23	0	0.19	0
SEPT Suboptimal	182	0.08	19.80	19.80	19.80	0.66	0.08	0.61	0.08
% Optimal*			0	0	0	0	0	0	0
% Best*		86.8	0	0	0	2.7	86.8	10.4	86.8
λ^{-1} = 60, Means									
SEPT Optimal	1129		48.93	38.22	0.24	0.27	0	0.24	0
SEPT Suboptimal	23	0.00	24.51	24.51	24.51	1.44	0.00	1.54	0.00
% Optimal*			0	0	0	0	0	0	0
% Best*		100	0	0	0	0	100	0	100

* Of the problems for which SEPT is suboptimal, the percentages for which the other rules are optimal and for which each rule finds the best heuristic solution.

Generally Decaying Rewards

As discussed in Section 6.4, our analysis of more general reward functions continues to focus on exponentially decaying rewards, but we no longer require the rewards to decay to zero or to decay at a uniform rate across jobs classes. Combining each of the parameter values presented in Table 17 yields a set of 4,320 test problems. These problems can no longer be solved to optimality using formulation (2), so our analysis focuses exclusively on the performance of the proposed heuristics. For these problems we consider each of the rules tested above (with the exception of $Rr\mu$, which was clearly outperformed by $R(r + \lambda)\mu$), in addition to the FPI heuristic paired with $R(r + \lambda)\mu$, TRI, and RTRI. The MLDS and RMLDS rules can just as easily be

paired with the FPI algorithm, but we do not include them in this analysis due to their extreme similarity with the TRI and RTRI rules. We also note that because RTRI and SEPT are again identical over all the tested problems (see Table 21), FPI+SEPT (not included) will be identical to FPI+RTRI. The performance of each heuristic is averaged over 2,500 simulation replications. While the computational effort for the three FPI-based rules is greater than for the non-fluid-based rules, the times were still quite manageable, averaging roughly 0.003 seconds per replication. In the absence of an optimal policy, the performance of a rule for a particular problem is measured as a percentage deviation from the rule with the best (highest) performance for that problem. Summary statistics for the percentage deviations for each rule are shown at the top of Table 21.

For these problems, the SEPT, TRI, RTRI, MLDS, and RMLDS rules continue to significantly outperform the $r\mu$-based heuristics. Among the non-fluid-based rules, TRI and MLDS show slightly better mean and worst-case performance than SEPT, RTRI, and RMLDS, but are less likely to be the best overall policy. However, in comparison with the non-fluid-based rules, the results in Table 21 point most strongly to the overall quality of the FPI heuristic, regardless of which underlying rule it uses for the fluid approximation. In agreement with the findings of Li and Glazebrook (2010), all three of the FPI pairings exhibit better mean and worst-case performance than their underlying rules, with the FPI+ $R(r+\lambda)\mu$ heuristic finding the best solution more often than any other rule. Recall that the FPI heuristic uses the fluid algorithm to approximate the value function in an approximate dynamic programming (DP) framework. As a result, we can view the quality of the different FPI rules as a reflection on the accuracy of the underlying approximations. It is clear that the approximate DP framework yields better solutions than the $r\mu$-, TRI-, and MLDS-based rules, but differences still exist between the FPI pairings.

On the full set of problems, FPI+RTRI has the best overall mean and worst-case deviation while FPI+$R(r+\lambda)\mu$ finds the largest percentage of best solutions. It is interesting to note that while TRI outperforms RTRI as a general purpose rule, the FPI+RTRI rule does significantly better than the FPI+TRI, indicating that using the RTRI rule gives a better approximation of the DP value function. Additionally, the

quality of the solutions found by using the $R(r+\lambda)\mu$ rule in the FPI heuristic suggests that $R(r+\lambda)\mu$ generates a good approximation of the value function, even though its general performance cannot compete with TRI or RTRI. Combined, either SEPT or FPI+ $R(r+\lambda)\mu$ finds the best overall solution in 88 percent of the original problems (not shown in the table). The fact that both $R(r+\lambda)\mu$ and RTRI on their own find larger percentages of best solutions than TRI indicates that the fluid approximation benefits from rules that frequently find the best solutions, rather than rules that merely perform well on average. This emphasis on the ability of the underlying rule to find best solutions is perhaps explained from the perspective of FPI's policy improvement component, which helps to smooth out the worst-case performances of the underlying rules.

Also presented in Table 21 are differences in the performance of the heuristics for each of the different reward pairings (initial rewards $R_j(0)$ and asymptotic rewards $R_j(\infty)$). Differences by the other problem parameters, which can be found in Table A-28 in the appendix, mirror the trends discussed for constant rewards in Table 19. Differences bases on just the initial reward values exhibit the same general behavior as in the constant reward problems. However, other meaningful differences emerge when looking at the different initial and asymptotic reward pairings. We can see that the $r\mu$-based rules perform the worst when there is a large difference between the initial rewards and the asymptotic rewards, while SEPT, TRI, RTRI, MLDS, and RMLDS improve in these scenarios. These results suggest that the both the initial reward values and the asymptotic reward values have significant impact on the performances of the various heuristics.

Table 21. Summary statistics for percentage deviations from the best heuristic and differences by reward structure for problems with generally decaying rewards

									FPI +		
		SEPT	$r\mu$	$R(r+\lambda)\mu$	TRI	RTRI	MLDS	RMLDS	$R(r+\lambda)\mu$	TRI	RTRI
Mean		0.41	9.05	4.27	0.38	0.41	0.37	0.42	0.31	0.36	0.20
Maximum		11.16	39.17	20.08	8.38	11.16	8.29	11.15	7.16	4.80	2.39
% Best*		56.0	15.3	26.0	18.8	56.0	21.4	55.7	59.7	18.2	41.6
$R_j(0)$	$R_j(\infty)$										
(0.98,0.8)	(0.9,0.6)	1.17	4.77	3.15	0.91	1.17	0.90	1.17	0.08	0.52	0.35
	(0.9,0.4)	0.71	5.72	3.94	0.62	0.71	0.61	0.71	0.34	0.45	0.27
	(0.9,0.2)	0.58	7.04	4.72	0.67	0.58	0.65	0.58	1.00	0.45	0.08
	(0.7,0.6)	0.70	7.66	3.73	0.44	0.70	0.43	0.69	0.03	0.44	0.36
	(0.7,0.4)	0.27	8.86	4.86	0.20	0.27	0.18	0.27	0.23	0.32	0.26
	(0.7,0.2)	0.15	10.42	5.85	0.29	0.15	0.27	0.18	0.87	0.24	0.06
(0.98,0.5)	(0.9,0.4)	0.14	9.09	3.57	0.09	0.14	0.08	0.14	0.05	0.45	0.25
	(0.9,0.2)	0.02	10.51	4.82	0.16	0.02	0.14	0.02	0.40	0.42	0.05
	(0.7,0.4)	0.09	12.99	3.16	0.07	0.09	0.05	0.09	0.05	0.25	0.19
	(0.7,0.2)	0.00	14.64	4.70	0.19	0.00	0.16	0.00	0.22	0.15	0.02
(0.9,0.8)	(0.7,0.6)	1.18	5.70	3.35	0.86	1.18	0.86	1.17	0.04	0.50	0.39
	(0.7,0.4)	0.60	6.78	4.40	0.48	0.60	0.47	0.60	0.26	0.38	0.28
	(0.7,0.2)	0.43	8.29	5.26	0.52	0.43	0.50	0.44	0.83	0.33	0.11
(0.9,0.5)	(0.7,0.4)	0.15	10.84	3.54	0.08	0.15	0.06	0.15	0.04	0.32	0.25
	(0.7,0.2)	0.01	12.49	4.99	0.15	0.01	0.13	0.01	0.14	0.23	0.05

* Percentage of problems for which each rule generates the best solution.

The FPI-based heuristics also show differences between the reward pairings. The FPI+TRI and FPI+RTRI heuristics show the same tendencies as the underlying TRI and RTRI rules, performing worst when the initial rewards are close together and the asymptotic rewards are still fairly high. In contrast, these heuristics (particularly FPI+RTRI) perform the best when the initial rewards are further apart and the drop-off in class 2's reward is large. The performance of the FPI+$R(r+\lambda)\mu$ heuristic exhibits the exact opposite trends from the other two FPI rules. Its performance is best when the rewards for the two classes show very little drop-off over time, while it performs worst when the drop-off in class 2's rewards is large. Comparing the performances of the three FPI pairings with SEPT shows that FPI+TRI and FPI+RTRI tend to perform well when SEPT does best and FPI+$R(r+\lambda)\mu$ tends to do well when SEPT does worst.

To dig further into this relationship with the performance of SEPT, we conclude our analysis with a look at how the various heuristics perform when SEPT does not produce the best solutions. Table 22 divides the problems according to whether SEPT finds the best solution and shows the mean percentage deviations within each group for the different heuristics. In addition, among the problems for which SEPT is not best, the table presents the percentage of problems for which each of the other rules finds the best solutions. Among these problems, we see that the $r\mu$-based rules continue to perform the worst while TRI and MLDS do better than their counterparts with rewards incorporated. Perhaps the most striking trend among these problems is the strength of the FPI+ $R(r+\lambda)\mu$ heuristic. The other two FPI heuristics perform worse when SEPT is not best while FPI+ $R(r+\lambda)\mu$ performs much better on this subset of problems, finding the best solution in over 70 percent of the problems and showing a mean deviation of less than 0.1 percent. In fact, of all the rules with mean deviations of less than one percent, the FPI+ $R(r+\lambda)\mu$ heuristic is the only one that actually does better when SEPT is not best.

Table 22. Mean percentage deviations from the best heuristic by the performance of SEPT, and percentage of problems for which SEPT is not best that other rules are best

									FPI +	
	N	SEPT	$r\mu$	$R(r+\lambda)\mu$	TRI	RTRI	MLDS	RMLDS	$R(r+\lambda)\mu$	TRI	RTRI
Mean											
SEPT Best	2378		11.79	4.17	0.07	0	0.06	0.00	0.50	0.28	0.06
SEPT Not Best	1901	0.94	5.56	4.40	0.78	0.94	0.76	0.94	0.06	0.47	0.37
% Best*			6.5	7.0	10.2	0	11.6	0.1	72.5	4.0	7.5

* Of the problems for which SEPT is not best, the percentage for which the other rules find the best solution.

6.6. Discussion

The design of the triage process is an important component of MCI response planning, and triage correspondingly receives a great deal of attention in the emergency medicine and trauma literature. Recent research has related the MCI triage problem to the more general problem of scheduling jobs in a single-server impatient jobs clearing system. This clearing system is marked by a large number of simultaneously generated jobs (patients), each requiring service (treatment). The jobs are impatient in that they will abandon the system (die) if forced to wait too long for service. Motivated by a consensus in the medical literature that (1) triage should incorporate patients' long-term survival prospects and (2) long-term, post-treatment survival probabilities decrease if treatment is delayed, we extend the existing research to include rewards (long-term survival probabilities) for service that diminish over time. The resulting "MCI triage" clearing system can be described as a single-server clearing system for impatient jobs with diminishing rewards. By extending the existing research to incorporate deteriorating survival probabilities, we aim to generate additional insights into how these probabilities impact triage decisions.

In general, our results confirm earlier findings that triage decisions should be made dynamically based on the number of patients waiting for treatment and on the treatment and survival profiles of these patients. However, our results show that using rules that include information about the patients' long-term survival probabilities leads to better system performance. In our analysis of problems with two jobs classes, with a range of problem sizes and patient profiles motivated by the medical literature, we find that in many cases it is preferable to prioritize less-critical patients (with

shorter expected treatment times and higher long-term survival probabilities) in spite of the fact that their survival probabilities deteriorate more slowly and they can survive much longer while waiting for treatment. In cases where the survival probabilities are more equal and the less-critical patients are capable of surviving for quite some time without treatment, we find that it can be preferable to prioritize the more critically wounded patients. The observation that these results sometimes run counter to the prioritizations implied by traditional triage categories highlights the subtleties and complexities involved in categorizing patients as either *immediate* or *delayed*. In reality, patients are distributed along a continuum of severity and determining the correct cutoffs between immediate and delayed treatment depends on a range of patient and population characteristics. In addition, our results provide some mathematical justification for the existence of the *expectant* category, since more severely wounded patients are often delayed in order to serve a larger number of less severely wounded patients.

Chapter 7. Summary and Conclusions

The research presented in this dissertation contributes in meaningful ways to the growing literature on applications of operations research models to problems in hospital management and public health administration. Our approaches to both the surgery scheduling problem and the mass casualty incident triage problem involve the use of mathematical models to develop new insights into previously understudied, yet practically important, aspects of the two problems. More generally, we demonstrate how exact and approximate stochastic dynamic programming algorithms can be used to study the processes of sequential decision-making under uncertainty that lie at the heart of these and many other problems in healthcare.

7.1. Contributions and Future Work

The Single-Day Surgery Scheduling Problem

Our study of the single-day surgery scheduling problem in Chapters 2 through 4 represents the first model to capture the sequential nature of an OR manager's daily request queue decisions as they relate to the underlying block schedule and block release policies. By explicitly incorporating the customer (i.e. surgeon and patient) satisfaction costs involved with these decisions into our model, we are able to show how OR managers can use threshold-based decision rules to balance between competing classes of demand for surgery. These intuitive threshold-based rules are optimal for the special case with a single operating room and unit durations, and our analysis shows that they can be extended to produce high-quality heuristics for more general problems with varying case durations and with multiple operating rooms.

While our approach to surgery scheduling is motivated by a case study of the scheduling system at the University of Maryland Medical Center, our analysis of the resulting model focuses on generating insight into more general settings. To this end, our computational results explore the sensitivity of the proposed thresholds to a range of important input parameters. In particular, the most important determinants of our proposed thresholds are the primary demand arrival patterns and the relative costs associated with deferral and blocking penalties. Empirical data collected from

hospitals' scheduling systems and OR suites, such as the data collected during our case study, can be used to estimate the demand arrival patterns, but the problem of eliciting appropriate deferral and blocking costs from OR stakeholders is ripe for future research. In addition, future research will use hospital data to set up a simulation environment incorporating other realistic components of the scheduling system (such as stochastic case durations, case cancellations, and swapping cases between ORs) and compare the performance of threshold-based schedules with actual schedules.

In closing, these threshold-based decision rules suggest a new way for hospitals and OR managers to look at block release policies. Traditionally, a block release date is a single day after which the OR manager may (if he chooses) release any remaining OR time to RQ cases. Alternatively, a threshold-based block release policy based on the research presented in this dissertation would release unused OR time gradually over the course of several days leading up to the day of surgery. Such a policy would be adaptable to differences in demand arrival patterns between specialties and to varying priority levels associated with different specialties. Furthermore, our threshold-based policies would maintain the transparency of traditional block release dates (the daily thresholds for each OR could be easily distributed), thus providing clear, empirical justification for an OR manager's decisions during the stochastic and often contentious development of single-day surgery schedules.

Mass Casualty Incident Triage

Our research into mass casualty incident response planning expands on existing research relating MCI triage to the general problem of scheduling impatient jobs in a single-server clearing system. The earlier research in this area failed to incorporate the important MCI and trauma concepts that (1) not all patients who receive treatment will ultimately survive and (2) long-term (post-treatment) survival probabilities decrease as patients are forced to wait for treatment. These concepts are particularly applicable in the aftermath of a MCI, where a large number of severely wounded patients are competing for limited medical resources, and are incorporated into the impatient jobs clearing system as diminishing rewards for service.

Our contributions to this more general problem, as presented in Chapter 6, are threefold. First, we show how two different SDP formulations from the literature can be extended and present a range of heuristic procedures that extend and improve upon earlier approaches. Second, we use the growing number of published studies on MCI response planning and casualty patterns, which rely on everything from expert opinion to statistical analyses and mathematical models, to generate a range of realistic problem instances for the MCI triage clearing system. Third, we perform extensive computational tests in order to gauge the quality of our heuristics and generate insights into the impact of deteriorating survival probabilities on MCI triage decisions. Our results not only confirm earlier findings that triage decisions should be made dynamically based on overall patient volume and individual patient characteristics, but they also indicate that including information about patients' long-term survival probabilities leads to better overall system performance.

Rather than provide definitive guidance on how triage decisions should be made in the aftermath of a MCI, the research in this dissertation lays the foundation for continued research that will seek to address many of the unresolved triage-related questions discussed in Chapter 5. Our computational results are limited by strong assumptions on the service and lifetime distributions and on the reward functions, and a first logical direction for future work is to test our heuristics under more general assumptions. In addition, our single-server model can easily be extended to problems with more than two job classes, which will provide further insight into which patient profiles are prioritized for immediate treatment in which scenarios. Obtaining additional data, either from existing trauma databases, analysis of previous MCIs, or hospital MCI response training exercises, will facilitate these continued efforts.

Other research on the problem of MCI triage will work to push our model beyond the single-server clearing system presented here. The single-server assumption does not allow for an analysis of how a finite amount of resources should be allocated between patient classes (for instance, should any of the ORs in a large OR suite be allocated to non-severely wounded patients?). Much of the medical literature in Chapter 5 advocates splitting the available resources into an "immediate" treatment area and a "delayed" treatment area, and a two-server model could be used to explore

the implications of this strategy (with the servers modeling the departure processes from each area). We also know that, in reality, patients' injuries lie along a continuum of injury severity and the extent of these injuries (and a patient's corresponding job class) cannot always be determined with certainty at the time of triage. For these reasons, an additional goal of our continued research in this area is to explore the appropriate cutoff points between immediate and delayed treatment along the continuum of injury severity and to test the sensitivity of these cutoffs to uncertainty in patients' true severity. Progress toward these goals will help bring mathematical insights to fundamental questions regarding the tradeoff between triage speed and triage accuracy (which is related to the question of who should perform triage), the decision to perform secondary triage, and the overall design of the triage process.

7.2. Broader Insights

In addition to generating new insights into the surgery scheduling and MCI triage problems, this dissertation sheds light on both the promise and difficulties involved in the application of operations research models, and particularly stochastic dynamic programs, to problems in healthcare. The healthcare system (including both the healthcare delivery system and the public health system) is faced with the twin challenges of complex, highly stochastic problems and a large number of stakeholders demanding intuitive, implementable solutions. Regardless of the modeling approach used, it is crucial to first identify and model only those components that contribute to the behavior being studied (model parsimony) and to then be able to relate the proposed solution algorithms back to some underlying intuition behind the problem (solution interpretability). These imperatives are doubly important for stochastic dynamic programs due to their inherent computational complexity.

In the sequential prioritization problems we study, the underlying intuition behind our solutions boils down to striking a balance between competing patient classes. Our threshold-based rules for the single-day surgery scheduling problem seek out a balance in the decision space between scheduling existing demand and preserving OR time for unknown future demand. For the MCI triage problem, this balance presents itself in the state space as a tipping point where a preference for one patient class

switches to a preference for a different class. In both cases, searching for this balance provides an intuitive framework from which to search for computationally efficient algorithms and from which to communicate and ultimately implement the resulting solutions. As future research works to apply stochastic dynamic programming to problems in healthcare, we feel that this notion of seeking balance in both the state and decision spaces, and the processes we use in this dissertation to identify shifts in this balance, can help ensure that operations researchers are finding meaningful, implementable solutions to the complex problems facing the healthcare industry.

Appendix

A.1. Supplementary Tables for Chapter 4

Table A-1. 95% confidence interval half-widths for mean percentage deviations presented in Table 13

$h_j^2 : h_j^1$					1:1				
$r_j^1 : h_j^1$		1:1			3:1			5:1	
$r_j^2 : r_j^1$	1:1	2:1	3:1	1:1	2:1	3:1	1:1	2:1	3:1
Greedy									
Duration	3.6	3.2	2.9	2.7	3.1	3.9	3.1	4.5	6.1
Ratios	3.8	3.4	3.0	2.7	3.2	4.0	3.1	4.5	6.2
Threshold First	3.8	3.4	3.0	2.7	3.2	4.0	3.2	4.5	6.1
Day-to-Day									
Duration	4.0	3.4	2.9	2.5	2.5	2.8	2.5	3.1	3.9
Ratios	4.5	3.7	3.2	2.7	2.7	2.9	2.7	3.2	3.9
Threshold First	4.0	3.4	2.9	2.5	2.5	2.8	2.5	3.1	3.9
Cumulative									
Duration	4.6	3.8	3.3	2.7	2.5	2.5	2.4	2.6	2.8
Ratios	5.1	4.3	3.6	3.0	2.7	2.7	2.7	2.8	3.1
Threshold First	4.6	3.8	3.2	2.6	2.5	2.4	2.4	2.6	2.8
Smart									
Duration	3.6	3.6	3.2	2.6	2.5	2.4	2.4	2.5	2.5
Ratios	3.8	3.7	3.3	2.9	2.7	2.7	2.7	2.8	2.8
Threshold First	3.8	3.7	3.2	2.6	2.5	2.4	2.4	2.4	2.5
$h_j^2 : h_j^1$ = 2:1					2:1				
Greedy									
Duration	3.5	2.9	2.9	2.3	2.5	3.0	2.5	3.4	4.7
Ratios	3.7	2.9	2.9	2.4	2.5	3.0	2.6	3.4	4.7
Threshold First	3.8	3.1	3.0	2.4	2.6	3.1	2.6	3.5	4.7
Day-to-Day									
Duration	4.4	3.5	3.2	2.3	2.2	2.3	2.2	2.5	3.1
Ratios	4.6	3.5	3.2	2.4	2.2	2.3	2.2	2.5	3.1
Threshold First	4.4	3.5	3.2	2.3	2.2	2.3	2.2	2.5	3.1
Cumulative									
Duration	5.5	4.3	3.8	2.7	2.4	2.3	2.3	2.3	2.5
Ratios	5.5	4.3	3.8	2.8	2.4	2.3	2.3	2.3	2.5
Threshold First	5.5	4.3	3.8	2.7	2.4	2.3	2.3	2.3	2.5
Smart									
Duration	3.5	2.9	3.3	2.5	2.3	2.3	2.2	2.3	2.3
Ratios	3.7	2.9	3.3	2.6	2.3	2.3	2.3	2.3	2.3
Threshold First	3.8	3.1	3.3	2.5	2.3	2.3	2.2	2.3	2.3

Table A-2. Mean percentage deviation from the optimal solution for three threshold-based heuristics applied to test problems representing a range of primary service line arrival patterns and cost structures (for $h_j^2 : h_j^1 = 1:1$ and $r_j^2 : r_j^1 = 1:1$)

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Greedy + Duration	6.2	10.0	30.4	1.3	19.6	61.5	5.2	21.4	64.2	9.0	13.5	42.7	0.9	17.3	62.1
Smart + Duration	15.2	18.2	9.3	1.3	8.3	2.9	12.3	7.8	1.6	22.4	9.8	8.0	7.3	4.5	4.7
Smart + Threshold First	18.8	18.3	9.6	18	7.2	1.8	27.2	6.8	0.3	33.2	8.5	6.7	13.9	3.3	3.7

Table A-3. Mean percentage deviation from the optimal solution for three threshold-based heuristics applied to test problems representing a range of primary service line arrival patterns and cost structures (for $h_j^2 : h_j^1 = 1:1$ and $r_j^2 : r_j^1 = 2:1$)

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Greedy + Duration	4.7	30.6	75.9	5.2	50.3	129.2	3.1	57.5	139.4	7.5	48.6	108.5	0.4	41.4	106.1
Smart + Duration	14.1	20.6	5.1	31.7	4.4	4.2	10.9	5.8	1.0	21.9	10.9	0.6	6.7	2.6	2.9
Smart + Threshold First	17.4	20.8	2.6	36.4	3.2	2.6	25.8	4.3	-0.4	32.5	9.0	-1.1	12.5	1.3	1.5

Table A-4. Mean percentage deviation from the optimal solution for three threshold-based heuristics applied to test problems representing a range of primary service line arrival patterns and cost structures (for $h_j^2 : h_j^1 = 1:1$ and $r_j^2 : r_j^1 = 3:1$)

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Greedy + Duration	9.4	52.9	112	7.9	87.2	194.9	8.1	97.4	215.1	10.1	87.1	176.6	6.9	64.1	157.5
Smart + Duration	45.9	9.0	2.3	20.8	3.7	3.8	14.3	0.7	0.5	24.9	7.9	1.7	26.3	2.2	4.9
Smart + Threshold First	47.6	6.1	0.3	20	2.2	2.3	16.7	-0.7	-0.9	25.5	6.2	-0.1	30.0	1.0	3.3

Table A-5. Mean percentage deviation from the optimal solution for three threshold-based heuristics applied to test problems representing a range of primary service line arrival patterns and cost structures (for $h_j^2 : h_j^1 = 2:1$ and $r_j^2 : r_j^1 = 1:1$)

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Greedy + Duration	6.7	3.1	17.5	-0.2	3	31.7	5.1	6.9	27.7	5.1	4.4	16.9	-1.0	4.4	29.5
Smart + Duration	6.7	23.6	8.6	-0.2	9.6	5.9	5.1	7.0	5.0	5.1	17.9	2.9	-1.0	7.7	4.6
Smart + Threshold First	20.3	25.7	9	25.3	9.7	6	32.7	8.9	5.0	21.4	20.0	2.9	18.6	10.0	4.6

Table A-6. Mean percentage deviation from the optimal solution for three threshold-based heuristics applied to test problems representing a range of primary service line arrival patterns and cost structures (for $h_j^2 : h_j^1 = 2:1$ and $r_j^2 : r_j^1 = 3:1$)

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Greedy + Duration	4.9	25.3	70.8	5.2	46.8	126.6	4.1	45.4	129.6	6.2	44.2	111.0	1.7	33.7	90.9
Smart + Duration	19.3	19.6	0.6	37.3	4.3	3.2	13.7	1.1	-0.5	24.0	9.3	-0.5	1.7	3.7	0.5
Smart + Threshold First	23.2	19.5	0.6	40	4.3	3.2	29.9	1.1	-0.5	36.3	9.3	-0.5	18.8	3.7	0.5

Table A-7. 95% confidence interval half-widths for mean percentage deviations presented in Table 14

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Greedy + Duration	4.8	3.8	4.3	2.9	2.5	3.4	2.2	1.8	2.5	2.9	2.4	3.0	3.5	2.8	3.6
Smart + Duration	4.8	3.9	3.7	3.5	2.3	2.3	2.2	1.7	1.6	2.9	2.1	2.1	3.5	2.6	2.6
Smart + Threshold First	5.0	3.9	3.7	3.8	2.3	2.3	2.4	1.7	1.6	3.1	2.1	2.1	3.7	2.6	2.6

Table A-8. 95% confidence interval half-widths for mean percentage deviations presented in Table A-2

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Greedy + Duration	5.0	3.9	4.1	3.6	2.7	3.1	2.4	2.0	2.4	3.3	2.5	2.8	3.5	3.0	3.7
Smart + Duration	5.2	4.1	3.7	3.6	2.6	2.4	2.5	1.9	1.8	3.4	2.3	2.2	3.6	2.9	2.9
Smart + Threshold First	5.3	4.2	3.7	3.8	2.6	2.4	2.5	1.9	1.7	3.4	2.3	2.2	3.7	2.8	2.9

Table A-9. 95% confidence interval half-widths for mean percentage deviations presented in Table A-3

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Greedy + Duration	4.9	4.2	5.3	3.2	3.1	4.5	2.3	2.3	3.4	3.2	2.9	3.9	3.4	3.3	4.7
Smart + Duration	5.1	4.1	3.5	3.6	2.5	2.5	2.4	1.8	1.7	3.3	2.2	2.1	3.5	2.7	2.8
Smart + Threshold First	5.1	4.2	3.4	3.7	2.5	2.4	2.5	1.8	1.7	3.4	2.2	2.0	3.6	2.7	2.8

Table A-10. 95% confidence interval half-widths for mean percentage deviations presented in Table A-4

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Greedy + Duration	4.8	5.0	6.8	2.9	3.9	6.1	2.2	3.0	4.8	2.9	3.8	5.5	3.6	4.0	6.1
Smart + Duration	5.4	3.6	3.2	3.2	2.4	2.5	2.3	1.7	1.7	2.9	2.3	2.1	3.9	2.7	2.8
Smart + Threshold First	5.4	3.5	3.2	3.2	2.4	2.5	2.3	1.7	1.7	2.9	2.3	2.0	4.0	2.7	2.8

Table A-11. 95% confidence interval half-widths for mean percentage deviations presented in Table A-5

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Greedy + Duration	5.3	3.7	3.7	3.5	2.3	2.5	2.4	1.8	1.8	3.1	2.3	2.3	3.3	2.6	2.9
Smart + Duration	5.3	4.1	3.5	3.5	2.5	2.2	2.4	1.8	1.7	3.1	2.5	2.0	3.3	2.7	2.6
Smart + Threshold First	5.5	4.1	3.5	3.8	2.5	2.2	2.6	1.8	1.7	3.3	2.5	2.0	3.7	2.8	2.6

Table A-12. 95% confidence interval half-widths for mean percentage deviations presented in Table A-6

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Greedy + Duration	4.5	4.1	5.6	2.9	3.0	4.7	2.1	2.2	3.4	2.8	2.9	4.1	3.3	3.2	4.6
Smart + Duration	4.7	3.9	3.2	3.3	2.3	2.3	2.2	1.6	1.6	3.0	2.2	1.9	3.3	2.6	2.6
Smart + Threshold First	4.8	3.9	3.2	3.3	2.3	2.3	2.3	1.6	1.6	3.1	2.2	1.9	3.6	2.6	2.6

Table A-13. Mean percentage deviation from the unconstrained optimal solution for three RQ policies paired with different block release dates across a range of primary service line arrival patterns and cost structures (for $h_j^2 : h_j^1 = 1:1$ and $r_j^2 : r_j^1 = 1:1$)

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Optimal Decisions															
Day 3	2.7	-4.2	1.1	-0.7	0.8	-0.3	-0.9	-0.6	1.1	-1.6	-0.9	-0.4	-2.5	-1.9	1.7
Day 2	24.0	1.8	1.6	20.4	5.5	-0.1	28.9	9.2	2.6	22.5	5.8	1.1	21.8	5.5	2.0
Day 1	60.4	15.3	6.2	44.8	10.4	1.0	56.5	12.7	2.7	51.8	15.1	3.1	55.6	12.0	3.7
Day 0	98.1	31.5	11.0	66.0	14.0	2.4	69.1	13.4	2.7	72.0	20.5	4.2	80.6	16.2	4.8
Greedy + Duration															
Day 3	7.5	8.6	36.9	4.1	20.0	64.1	3.3	19.5	67.1	5.5	15.7	44.4	-0.2	16.0	62.7
Day 2	28.4	6.2	13.1	27.2	19.7	42.9	36.7	36.0	75.2	34.3	23.0	38.0	26.2	22.5	53.5
Day 1	66.6	20.5	13.2	51.6	18.1	21.9	67.4	37.3	54.5	62.5	27.7	26.2	62.0	28.8	42.4
Day 0	103.5	35.1	13.9	69.2	16.3	4.5	71.3	15.0	4.1	74.9	22.6	6.0	84.7	18.6	7.0
Smart + Duration															
Day 3	17.0	14.7	11.5	29.9	8.2	3.1	11.2	7.8	4.1	19.4	9.8	8.2	7.2	4.6	4.6
Day 2	28.4	22.6	13.0	51.8	12.0	3.1	51.9	15.0	4.1	66.5	22.4	8.2	26.2	9.2	4.6
Day 1	66.6	32.4	17.3	67.0	13.4	3.4	65.8	15.0	4.1	68.6	23.7	8.2	62.0	15.9	6.4
Day 0	103.5	35.1	13.9	69.2	16.3	4.5	71.3	15.0	4.1	74.9	22.6	6.0	84.7	18.6	7.0

Table A-14. Mean percentage deviation from the unconstrained optimal solution for three RQ policies paired with different block release dates across a range of primary service line arrival patterns and cost structures (for $h_j^2 : h_j^1 = 1:1$ and $r_j^2 : r_j^1 = 2:1$)

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Optimal Decisions															
Day 3	0.4	1.5	-1.1	0.2	-1.6	0.4	0.3	0.1	-0.3	0.2	1.5	0.2	-2.4	-1.4	-0.1
Day 2	21.4	3.5	-0.9	20.3	-0.4	0.4	30.5	2.5	-0.3	24.2	2.5	0.2	21.6	2.3	-0.1
Day 1	56.4	11.5	2.2	44.5	1.4	0.4	57.8	2.8	-0.3	52.2	5.8	0.2	56.3	5.1	0.9
Day 0	92.1	20.9	4.8	65.5	3.6	0.6	71.0	2.8	-0.3	71.0	9.9	0.9	81.2	7.7	2.3
Greedy + Duration															
Day 3	5.8	32.5	71.7	5.3	50.2	130.4	4.2	56.5	139.6	6.4	49.6	109.1	-0.6	41.6	107.4
Day 2	27.3	11.4	25.1	26.9	33.5	87.6	38.1	65.4	136.8	35.0	48.5	91.1	25.8	30.0	73.5
Day 1	62.5	18.2	15.0	50.8	16.3	38.5	68.9	48.0	85.9	62.9	35.7	53.6	63.0	27.2	48.1
Day 0	97.0	24.1	7.7	68.6	5.7	2.5	73.5	4.3	1.0	74.0	11.6	2.6	85.1	10.1	4.4
Smart + Duration															
Day 3	15.3	22.5	1.8	32.4	3.1	4.1	11.5	4.3	1.0	20.2	11.6	2.6	6.2	2.0	3.0
Day 2	27.3	24.1	1.8	53.9	3.1	4.1	51.8	4.3	1.0	66.8	11.6	2.6	25.8	4.4	3.0
Day 1	62.5	28.6	4.4	68.6	3.5	4.1	67.0	4.3	1.0	67.4	11.6	2.6	63.0	7.9	4.1
Day 0	97.0	24.1	7.7	68.6	5.7	2.5	73.5	4.3	1.0	74.0	11.6	2.6	85.1	10.1	4.4

Table A-15. Mean percentage deviation from the unconstrained optimal solution for three RQ policies paired with different block release dates across a range of primary service line arrival patterns and cost structures (for $h_j^2 : h_j^1 = 1:1$ and $r_j^2 : r_j^1 = 3:1$)

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Optimal Decisions															
Day 3	0.9	-1.8	0.1	-0.2	-1.3	0.1	1.5	-0.9	-0.8	-0.3	-0.6	0.0	1.7	-1.1	0.5
Day 2	13.4	0.2	0.1	13.6	-1.0	0.1	20.5	-0.8	-0.8	15.5	-0.4	0.0	18.1	0.1	0.5
Day 1	39.5	6.1	1.3	29.1	-0.6	0.1	37.5	-0.8	-0.8	32.8	1.9	0.0	44.1	2.0	0.6
Day 0	68.8	13.5	3.6	44.3	0.4	1.1	43.4	-0.8	-0.8	45.3	5.3	0.6	62.1	4.3	1.9
Greedy + Duration															
Day 3	9.6	52.5	109.6	11.7	89.9	197.0	9.8	98.9	215.8	11.7	84.8	184.0	6.9	68.2	157.5
Day 2	20.5	18.2	43.7	23.5	56.3	123.9	34.9	99.5	203.6	31.3	77.6	152.5	22.5	41.8	98.9
Day 1	46.0	14.5	22.8	35.7	24.3	50.9	52.4	64.4	119.3	47.5	48.9	86.1	50.7	29.0	59.4
Day 0	73.1	16.9	6.4	46.9	2.3	3.0	45.5	0.6	0.5	47.7	7.1	2.2	65.6	6.6	4.1
Smart + Duration															
Day 3	45.3	8.3	3.9	22.4	3.1	3.6	15.7	0.6	0.5	26.9	7.1	2.2	28.0	3.4	4.7
Day 2	55.9	8.3	3.9	37.5	3.1	3.6	30.4	0.6	0.5	34.3	7.1	2.2	39.3	3.4	4.7
Day 1	68.8	12.8	3.8	50.7	3.1	3.6	46.7	0.6	0.5	49.3	7.1	2.2	60.2	4.8	4.7
Day 0	73.1	16.9	6.4	46.9	2.3	3.0	45.5	0.6	0.5	47.7	7.1	2.2	65.6	6.6	4.1

Table A-16. Mean percentage deviation from the unconstrained optimal solution for three RQ policies paired with different block release dates across a range of primary service line arrival patterns and cost structures (for $h_j^2 : h_j^1 = 2:1$ and $r_j^2 : r_j^1 = 1:1$)

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Optimal Decisions															
Day 3	3.3	-1.6	-0.9	0.5	-0.5	1.9	0.6	2.2	0.4	4.1	0.4	-1.4	0.3	2.3	-0.5
Day 2	41.0	9.7	2.1	41.2	12.5	3.4	50.3	24.4	6.1	45.0	17.1	5.0	44.7	18.6	3.7
Day 1	92.0	33.8	13.1	84.9	24.6	6.0	104.3	36.0	6.9	93.5	38.0	9.8	99.3	34.4	8.1
Day 0	142.7	56.9	24.0	118.1	34.1	8.5	129.9	38.3	6.9	124.7	47.1	11.8	139.2	44.3	11.4
Greedy + Duration															
Day 3	6.9	3.0	17.8	1.7	4.9	32.0	3.1	6.6	27.7	10.2	7.0	15.9	0.4	8.1	28.0
Day 2	42.8	12.2	6.0	43.9	17.2	26.5	53.3	31.7	41.9	53.8	24.7	20.3	44.8	26.1	28.9
Day 1	92.8	35.6	15.3	86.5	27.8	16.4	107.7	48.3	37.4	99.4	42.5	21.5	100.1	44.5	29.9
Day 0	142.7	56.9	24.0	118.1	34.1	8.5	129.9	38.3	6.9	124.7	47.1	11.8	139.2	44.3	11.4
Smart + Duration															
Day 3	6.9	22.3	9.5	1.7	11.5	6.5	3.1	7.0	4.5	10.2	18.5	2.1	0.4	11.6	5.6
Day 2	42.8	34.6	14.0	43.9	25.4	7.1	53.3	26.4	6.9	53.8	40.4	11.7	44.8	22.4	7.9
Day 1	92.8	52.2	22.1	86.5	33.5	7.1	107.7	39.6	6.9	99.4	42.8	12.9	100.1	39.2	10.0
Day 0	142.7	56.9	24.0	118.1	34.1	8.5	129.9	38.3	6.9	124.7	47.1	11.8	139.2	44.3	11.4

Table A-17. Mean percentage deviation from the unconstrained optimal solution for three RQ policies paired with different block release dates across a range of primary service line arrival patterns and cost structures (for $h_j^2 : h_j^1 = 2:1$ and $r_j^2 : r_j^1 = 3:1$)

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Optimal Decisions															
Day 3	-3.8	-0.7	-1.0	0.4	-0.5	0.3	0.0	-0.1	0.7	-0.3	0.7	0.7	0.8	-0.4	2.2
Day 2	23.4	2.9	-1.0	29.8	0.2	0.3	36.2	1.2	0.7	31.3	2.1	0.7	34.7	4.0	2.1
Day 1	64.6	10.8	2.0	59.0	1.9	0.3	73.4	1.2	0.7	64.9	4.5	0.7	78.1	9.6	2.5
Day 0	106.1	20.8	5.2	81.9	4.3	1.2	85.6	1.2	0.7	83.8	7.6	1.2	109.3	13.5	4.7
Greedy + Duration															
Day 3	1.1	26.2	65.1	5.2	47.8	126.2	3.0	45.8	129.7	7.4	45.2	111.8	2.8	35.4	98.7
Day 2	26.3	8.3	18.9	33.5	31.6	81.0	40.7	55.4	127.1	40.1	46.4	95.5	34.9	22.6	60.9
Day 1	66.7	13.1	10.0	60.8	14.1	33.4	77.2	39.5	81.0	70.0	32.6	55.9	79.0	23.1	39.7
Day 0	106.1	20.8	5.2	81.9	4.3	1.2	85.6	1.2	0.7	83.8	7.6	1.2	109.3	13.5	4.7
Smart + Duration															
Day 3	14.4	20.0	-1.0	36.4	3.2	2.5	12.1	1.2	0.7	25.9	10.2	1.2	2.8	4.1	4.2
Day 2	26.3	20.5	-1.0	63.0	3.2	2.5	59.9	1.2	0.7	78.6	10.2	1.2	34.9	4.2	4.2
Day 1	66.7	25.0	1.8	80.0	2.0	2.5	78.5	1.2	0.7	94.6	10.2	1.2	79.0	10.1	4.8
Day 0	106.1	20.8	5.2	81.9	4.3	1.2	85.6	1.2	0.7	83.8	7.6	1.2	109.3	13.5	4.7

Table A-18. 95% confidence interval half-widths for the mean percentage deviations presented in Table 15

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Optimal Decisions															
Day 3	4.6	3.4	3.2	2.9	2.1	2.2	2.2	1.6	1.6	3.0	2.1	2.0	3.5	2.4	2.6
Day 2	5.1	3.5	3.2	3.3	2.3	2.2	2.6	1.8	1.6	3.3	2.2	2.0	4.1	2.6	2.6
Day 1	6.1	3.8	3.3	4.0	2.4	2.2	3.3	1.8	1.6	3.9	2.2	2.0	5.1	2.8	2.6
Day 0	7.3	4.1	3.3	4.7	2.4	2.3	3.7	1.8	1.6	4.3	2.3	2.0	6.1	2.9	2.6
Greedy + Duration															
Day 3	4.8	3.8	4.3	3.1	2.5	3.4	2.3	1.9	2.5	3.2	2.4	3.0	3.5	2.7	3.6
Day 2	5.2	3.7	3.7	3.5	2.6	3.3	2.7	2.1	2.7	3.5	2.6	3.1	4.1	2.8	3.6
Day 1	6.2	3.9	3.6	4.1	2.6	3.0	3.4	2.3	2.7	4.0	2.6	2.9	5.1	3.1	3.5
Day 0	7.3	4.1	3.3	4.7	2.4	2.3	3.7	1.8	1.6	4.3	2.3	2.0	6.1	2.9	2.6
Smart + Duration															
Day 3	4.8	3.9	3.6	3.1	2.3	2.4	2.3	1.8	1.6	3.2	2.1	2.1	3.5	2.5	2.6
Day 2	5.2	4.1	3.6	3.5	2.4	2.4	2.7	1.8	1.6	3.5	2.3	2.1	4.1	2.6	2.6
Day 1	6.2	4.3	3.7	4.1	2.4	2.3	3.4	1.8	1.6	4.0	2.3	2.1	5.1	2.8	2.6
Day 0	7.3	4.1	3.3	4.7	2.4	2.3	3.7	1.8	1.6	4.3	2.3	2.0	6.1	2.9	2.6

Table A-19. 95% confidence interval half-widths for the mean percentage deviations presented in Table A-13

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Optimal Decisions															
Day 3	4.8	3.5	3.3	2.9	2.4	2.3	2.2	1.7	1.8	2.7	2.4	2.2	3.4	2.7	2.7
Day 2	5.1	3.6	3.3	3.1	2.4	2.3	2.3	1.8	1.8	2.8	2.4	2.1	3.7	2.8	2.7
Day 1	5.6	3.9	3.3	3.5	2.5	2.3	2.7	1.9	1.8	3.2	2.5	2.1	4.2	2.9	2.7
Day 0	6.3	4.2	3.4	3.8	2.6	2.4	2.9	1.9	1.8	3.4	2.5	2.1	4.8	3.1	2.7
Greedy + Duration															
Day 3	5.1	3.8	4.1	3.1	2.6	3.2	2.3	1.9	2.4	3.0	2.6	2.8	3.5	2.9	3.7
Day 2	5.3	3.8	3.8	3.4	2.7	3.2	2.5	2.1	2.6	3.2	2.7	2.8	3.8	3.2	3.8
Day 1	5.8	4.0	3.7	3.7	2.8	2.9	2.9	2.3	2.6	3.5	2.8	2.7	4.5	3.4	3.7
Day 0	6.5	4.3	3.5	3.9	2.7	2.4	2.9	2.0	1.8	3.5	2.5	2.2	4.9	3.2	2.8
Smart + Duration															
Day 3	5.2	4.0	3.7	3.5	2.5	2.4	2.4	1.9	1.8	3.1	2.4	2.3	3.6	2.9	2.8
Day 2	5.3	4.1	3.7	3.7	2.6	2.4	2.8	2.0	1.8	3.6	2.6	2.3	3.8	3.0	2.8
Day 1	5.8	4.4	3.8	4.0	2.6	2.4	2.8	2.0	1.8	3.4	2.6	2.3	4.5	3.1	2.8
Day 0	6.5	4.3	3.5	3.9	2.7	2.4	2.9	2.0	1.8	3.5	2.5	2.2	4.9	3.2	2.8

Table A-20. 95% confidence interval half-widths for the mean percentage deviations presented in Table A-14

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Optimal Decisions															
Day 3	4.6	3.6	3.3	3.0	2.4	2.3	2.2	1.7	1.7	2.9	2.4	2.0	3.4	2.7	2.7
Day 2	4.8	3.6	3.3	3.1	2.4	2.3	2.4	1.7	1.7	3.0	2.3	2.0	3.6	2.7	2.7
Day 1	5.3	3.7	3.3	3.5	2.4	2.3	2.7	1.7	1.7	3.3	2.3	2.0	4.2	2.8	2.7
Day 0	6.0	3.8	3.4	3.8	2.4	2.3	2.9	1.7	1.7	3.5	2.3	2.0	4.8	2.8	2.8
Greedy + Duration															
Day 3	4.9	4.3	5.3	3.2	3.1	4.4	2.3	2.3	3.4	3.2	3.0	3.9	3.4	3.3	4.7
Day 2	5.1	4.0	4.5	3.4	3.1	4.2	2.6	2.5	3.6	3.4	3.1	4.0	3.8	3.3	4.5
Day 1	5.5	4.0	4.1	3.7	2.9	3.5	3.0	2.5	3.4	3.6	3.0	3.6	4.4	3.4	4.1
Day 0	6.2	3.9	3.5	3.9	2.5	2.3	3.0	1.8	1.7	3.6	2.3	2.1	4.9	2.9	2.8
Smart + Duration															
Day 3	5.1	4.2	3.4	3.6	2.5	2.4	2.4	1.8	1.7	3.2	2.3	2.1	3.5	2.8	2.8
Day 2	5.1	4.2	3.4	3.8	2.5	2.4	2.8	1.8	1.7	3.7	2.3	2.1	3.8	2.8	2.8
Day 1	5.5	4.3	3.4	4.1	2.5	2.4	2.9	1.8	1.7	3.4	2.3	2.1	4.4	2.9	2.8
Day 0	6.2	3.9	3.5	3.9	2.5	2.3	3.0	1.8	1.7	3.6	2.3	2.1	4.9	2.9	2.8

Table A-21. 95% confidence interval half-widths for the mean percentage deviations presented in Table A-15

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Optimal Decisions															
Day 3	4.5	3.5	3.2	2.9	2.3	2.4	2.1	1.7	1.7	2.6	2.2	2.0	3.3	2.7	2.7
Day 2	4.5	3.5	3.2	3.0	2.3	2.4	2.2	1.7	1.7	2.7	2.2	2.0	3.5	2.8	2.7
Day 1	4.9	3.6	3.2	3.2	2.3	2.4	2.4	1.7	1.7	2.9	2.2	2.0	4.0	2.8	2.7
Day 0	5.4	3.6	3.2	3.4	2.3	2.4	2.4	1.7	1.7	2.9	2.1	2.0	4.3	2.8	2.7
Greedy + Duration															
Day 3	4.7	5.1	6.7	3.1	4.0	6.1	2.2	3.0	4.7	2.9	3.7	5.5	3.5	4.1	6.2
Day 2	4.8	4.4	5.4	3.3	3.7	5.5	2.4	3.1	4.8	3.1	3.8	5.5	3.7	3.9	5.4
Day 1	5.1	4.0	4.6	3.4	3.2	4.2	2.7	3.0	4.4	3.2	3.5	4.7	4.2	3.6	4.6
Day 0	5.6	3.7	3.3	3.5	2.4	2.4	2.5	1.7	1.7	3.0	2.2	2.1	4.5	2.9	2.8
Smart + Duration															
Day 3	5.3	3.5	3.3	3.3	2.4	2.5	2.2	1.7	1.7	2.9	2.2	2.1	3.9	2.9	2.9
Day 2	5.4	3.5	3.3	3.5	2.4	2.5	2.3	1.7	1.7	2.8	2.2	2.1	4.1	2.9	2.9
Day 1	5.6	3.6	3.3	3.7	2.4	2.5	2.5	1.7	1.7	3.1	2.2	2.1	4.4	2.9	2.9
Day 0	5.6	3.7	3.3	3.5	2.4	2.4	2.5	1.7	1.7	3.0	2.2	2.1	4.5	2.9	2.8

Table A-22. 95% confidence interval half-widths for the mean percentage deviations presented in Table A-16

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Optimal Decisions															
Day 3	4.9	3.4	3.3	3.0	2.2	2.2	2.2	1.7	1.6	3.1	2.3	2.0	3.6	2.7	2.5
Day 2	5.3	3.6	3.3	3.5	2.5	2.3	2.6	2.0	1.7	3.4	2.5	2.1	4.2	3.0	2.7
Day 1	6.2	4.2	3.5	4.2	2.7	2.4	3.3	2.2	1.7	4.0	2.8	2.1	5.2	3.4	2.8
Day 0	7.4	4.8	3.8	4.8	3.0	2.4	3.7	2.2	1.7	4.4	2.9	2.2	6.1	3.7	2.8
Greedy + Duration															
Day 3	5.1	3.6	3.7	3.1	2.3	2.6	2.3	1.7	1.8	3.3	2.4	2.3	3.6	2.8	2.9
Day 2	5.4	3.8	3.5	3.6	2.6	2.7	2.7	2.0	2.0	3.6	2.7	2.4	4.2	3.2	3.2
Day 1	6.3	4.2	3.7	4.2	2.8	2.6	3.4	2.4	2.2	4.1	2.9	2.5	5.2	3.6	3.3
Day 0	7.4	4.8	3.8	4.8	3.0	2.4	3.7	2.2	1.7	4.4	2.9	2.2	6.1	3.7	2.8
Smart + Duration															
Day 3	5.1	4.0	3.6	3.1	2.5	2.3	2.3	1.8	1.6	3.3	2.5	2.0	3.6	2.9	2.7
Day 2	5.4	4.2	3.7	3.6	2.8	2.4	2.7	2.0	1.7	3.6	2.9	2.2	4.2	3.1	2.8
Day 1	6.3	4.7	3.9	4.2	3.0	2.4	3.4	2.3	1.7	4.1	2.8	2.2	5.2	3.5	2.8
Day 0	7.4	4.8	3.8	4.8	3.0	2.4	3.7	2.2	1.7	4.4	2.9	2.2	6.1	3.7	2.8

Table A-23. 95% confidence interval half-widths for the mean percentage deviations presented in Table A-17

	Early, Early			Middle, Middle			Late, Late			Early, Late			Late, Early		
$r_j^1 : h_j^1$	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1	1:1	3:1	5:1
Optimal Decisions															
Day 3	4.1	3.5	3.1	2.7	2.2	2.2	1.9	1.6	1.7	2.6	2.1	2.0	3.2	2.5	2.6
Day 2	4.4	3.6	3.1	3.1	2.2	2.2	2.3	1.6	1.7	2.9	2.1	2.0	3.7	2.7	2.6
Day 1	5.2	3.7	3.1	3.6	2.2	2.2	2.8	1.6	1.7	3.4	2.1	2.0	4.5	2.8	2.6
Day 0	6.2	3.7	3.2	4.0	2.3	2.2	3.0	1.6	1.7	3.6	2.1	2.0	5.2	2.9	2.6
Greedy + Duration															
Day 3	4.3	4.2	5.4	2.9	3.1	4.6	2.0	2.2	3.5	2.9	2.9	4.1	3.3	3.3	4.7
Day 2	4.6	3.9	4.4	3.2	3.0	4.3	2.4	2.4	3.6	3.1	3.0	4.2	3.7	3.2	4.2
Day 1	5.3	3.8	3.9	3.7	2.7	3.5	2.9	2.4	3.4	3.5	2.9	3.7	4.5	3.2	3.8
Day 0	6.2	3.7	3.2	4.0	2.3	2.2	3.0	1.6	1.7	3.6	2.1	2.0	5.2	2.9	2.6
Smart + Duration															
Day 3	4.5	4.0	3.1	3.4	2.3	2.3	2.1	1.6	1.7	3.1	2.2	2.0	3.3	2.7	2.7
Day 2	4.6	4.0	3.1	3.7	2.3	2.3	2.7	1.6	1.7	3.6	2.2	2.0	3.7	2.7	2.7
Day 1	5.3	4.1	3.1	4.1	2.2	2.3	2.9	1.6	1.7	4.0	2.2	2.0	4.5	2.8	2.7
Day 0	6.2	3.7	3.2	4.0	2.3	2.2	3.0	1.6	1.7	3.6	2.1	2.0	5.2	2.9	2.6

A.2. Supplementary Tables for Chapter 6

Table A-24. Mean percentage deviations from the performance of the optimal policy for different versions of the fluid-based policy improvement heuristic for problems with $\lambda = 0$ by different input parameters

	Rrμ	Modified FPI+Rrμ	Original FPI+*Rrμ**	Original FPI+*Rrμ***
Total	2.04	0.04	0.59	0.33
(n_1, n_2)				
(10,5)	1.68	-0.05	0.36	0.24
(15,5)	1.81	0.16	0.62	0.39
(15,10)	2.64	0.00	0.78	0.36
(r_1^{-1}, r_2^{-1})				
(240,60)	3.67	0.07	0.56	0.15
(960,120)	0.42	0.00	0.61	0.51
(μ_1^{-1}, μ_2^{-1})				
(5,20)	0.41	0.00	1.10	0.47
(10,25)	1.41	0.14	0.23	0.24
(15,30)	4.31	-0.04	0.43	0.28
(R_1, R_2)				
(0.9,0.8)	2.57	0.14	0.35	0.28
(0.9,0.5)	1.51	-0.07	0.82	0.38

Note: Numerical integration parameters for the original FPI set to time-step Δt and stopping criterion ε.
* Δt = 0.5 and ε = 0.001
** Δt = 0.25 and ε = 0.0005

Table A-25. Computation times per replication for different versions of the fluid-based policy improvement heuristic for problems with $\lambda = 0$ by different input parameters

	Rrμ	Modified FPI+Rrμ	Original FPI+*Rrμ**	Original FPI+*Rrμ***
Total	<0.001s	<0.001s	5.29s	16.07s
(n_1, n_2)				
(10,5)	<0.001	<0.001	1.73	3.99
(15,5)	<0.001	<0.001	4.33	16.17
(15,10)	<0.001	<0.001	9.82	28.04
(r_1^{-1}, r_2^{-1})				
(240,60)	<0.001	<0.001	4.82	14.38
(960,120)	<0.001	<0.001	5.76	17.75
(μ_1^{-1}, μ_2^{-1})				
(5,20)	<0.001	<0.001	4.99	17.03
(10,25)	<0.001	<0.001	5.25	15.15
(15,30)	<0.001	<0.001	5.64	16.02
($R_1(0)$, $R_2(0)$)				
(0.9,0.8)	<0.001	<0.001	5.56	16.84
(0.9,0.5)	<0.001	<0.001	5.02	15.29

Note: Numerical integration parameters for the original FPI set to time-step Δt and stopping criterion ε.
* $\Delta t = 0.5$ and $\varepsilon = 0.001$
** $\Delta t = 0.25$ and $\varepsilon = 0.0005$

Table A-26. Mean percentage deviations from optimal policy for non-fluid-based heuristics for problems with uniformly decaying rewards ($\lambda^{-1} = 180$) by different input parameters

	SEPT	$r\mu$	$Rr\mu$	$R(r+\lambda)\mu$	TRI	RTRI	MLDS	RMLDS
(n_1, n_2)								
(20,10)	0.02	27.26	20.71	3.30	0.36	0.02	0.34	0.02
(20,15)	0.02	30.94	23.54	3.77	0.27	0.02	0.26	0.02
(30,10)	0.00	28.50	21.83	3.70	0.30	0.00	0.24	0.00
(30,20)	0.01	34.95	26.81	4.56	0.25	0.01	0.21	0.01
(r_1^{-1}, r_2^{-1})								
(240,60)	0.01	25.93	16.40	3.30	0.16	0.01	0.12	0.01
(480,60)	0.02	33.09	27.85	6.49	0.36	0.02	0.30	0.02
(720,60)	0.03	31.68	31.68	7.88	0.59	0.03	0.51	0.03
(960,60)	0.03	30.96	30.96	8.59	0.79	0.03	0.72	0.03
(240,120)	0	15.52	1.69	0	0.03	0	0.02	0
(480,120)	0.00	28.73	18.39	0.85	0.08	0.00	0.08	0.00
(720,120)	0.00	39.08	26.42	1.44	0.14	0.00	0.13	0.00
(960,120)	0.01	38.30	32.39	2.10	0.19	0.01	0.19	0.01
(μ_1^{-1}, μ_2^{-1})								
(5,20)	0	34.03	20.46	0	0.05	0	0.05	0
(5,25)	0	27.04	16.36	0	0.04	0	0.04	0
(5,30)	0	29.93	16.70	0	0.05	0	0.04	0
(10,20)	0.01	29.88	25.25	4.49	0.24	0.01	0.18	0.01
(10,25)	0.00	31.39	25.88	4.17	0.20	0.00	0.15	0.00
(10,30)	0	36.17	30.02	0	0.18	0	0.15	0
(15,20)	0.07	21.06	18.91	10.34	0.97	0.07	0.97	0.07
(15,25)	0.02	29.00	25.43	9.92	0.49	0.02	0.44	0.02
(15,30)	0.01	35.20	29.98	5.56	0.43	0.01	0.30	0.01
(R_1, R_2)								
(0.98,0.8)	0.01	26.82	23.02	5.70	0.18	0.01	0.15	0.01
(0.98,0.5)	0.00	35.70	24.38	1.51	0.45	0.00	0.41	0.00
(0.9,0.8)	0.03	24.76	22.14	6.30	0.14	0.03	0.11	0.03
(0.9,0.5)	0.00	34.35	23.35	1.82	0.41	0.00	0.37	0.00

Table A-27. Mean percentage deviations from optimal policy for non-fluid-based heuristics for problems uniformly decaying rewards ($\lambda^{-1} = 60$) by different input parameters

	SEPT	$r\mu$	$Rr\mu$	$R(r+\lambda)\mu$	TRI	RTRI	MLDS	RMLDS
(n_1, n_2)								
(20,10)	0.00	46.21	36.16	0.68	0.46	0.00	0.44	0.00
(20,15)	0.00	49.15	38.51	0.73	0.30	0.00	0.30	0.00
(30,10)	0.00	46.92	36.72	0.69	0.26	0.00	0.20	0.00
(30,20)	0.00	51.47	40.38	0.78	0.17	0.00	0.14	0.00
(r_1^{-1}, r_2^{-1})								
(240,60)	0.00	39.70	25.53	0.63	0.16	0.00	0.12	0.00
(480,60)	0.00	54.94	46.80	1.41	0.39	0.00	0.34	0.00
(720,60)	0.00	54.82	54.82	1.41	0.65	0.00	0.60	0.00
(960,60)	0.00	54.81	54.81	2.30	0.91	0.00	0.86	0.00
(240,120)	0	21.53	2.50	0	0.01	0	0.01	0
(480,120)	0	42.89	27.69	0	0.05	0	0.04	0
(720,120)	0	59.43	40.73	0	0.08	0	0.08	0
(960,120)	0	59.38	50.65	0	0.12	0	0.12	0
(μ_1^{-1}, μ_2^{-1})								
(5,20)	0	56.41	35.55	0	0.07	0	0.07	0
(5,25)	0	43.38	27.12	0	0.05	0	0.05	0
(5,30)	0	45.70	26.47	0	0.06	0	0.05	0
(10,20)	0	50.41	43.81	0	0.28	0	0.23	0
(10,25)	0	50.03	41.94	0	0.21	0	0.16	0
(10,30)	0	54.39	45.76	0	0.15	0	0.13	0
(15,20)	0.00	37.41	34.24	5.46	1.11	0.00	1.14	0.00
(15,25)	0	46.03	41.15	1.02	0.43	0	0.39	0
(15,30)	0	52.18	45.43	0	0.32	0	0.22	0
(R_1, R_2)								
(0.98,0.8)	0.00	43.45	37.90	1.15	0.24	0.00	0.21	0.00
(0.98,0.5)	0	55.82	39.32	0	0.39	0	0.35	0
(0.9,0.8)	0.00	40.51	36.68	1.73	0.20	0.00	0.18	0.00
(0.9,0.5)	0	53.97	37.87	0	0.36	0	0.33	0

Table A-28. Mean percentage deviations from the best overall heuristic for problems with generally decaying rewards by different input parameters

								FPI +		
	SEPT	$r\mu$	$R(r+\lambda)\mu$	TRI	RTRI	MLDS	RMLDS	$R(r+\lambda)\mu$	TRI	RTRI
(n_1, n_2)										
(20,10)	0.58	7.38	3.46	0.52	0.58	0.50	0.58	0.24	0.22	0.40
(20,15)	0.59	9.12	4.04	0.53	0.59	0.51	0.59	0.24	0.21	0.44
(30,10)	0.22	8.29	4.26	0.23	0.22	0.21	0.22	0.36	0.18	0.27
(30,20)	0.27	11.42	5.31	0.25	0.27	0.24	0.28	0.39	0.19	0.34
(r_1^{-1}, r_2^{-1})										
(240,60)	0.04	13.60	6.41	0.15	0.04	0.11	0.04	0.42	0.13	0.18
(480,60)	0.30	10.40	6.22	0.29	0.30	0.27	0.30	0.14	0.22	0.29
(720,60)	0.69	6.50	4.66	0.57	0.69	0.55	0.69	0.09	0.23	0.31
(960,60)	1.10	4.53	3.52	0.89	1.10	0.86	1.10	0.07	0.24	0.31
(240,120)	0.00	9.23	2.36	0.04	0.00	0.03	0.01	0.81	0.09	0.23
(480,120)	0.10	10.25	4.00	0.12	0.10	0.12	0.11	0.52	0.17	0.47
(720,120)	0.37	10.34	3.72	0.35	0.37	0.35	0.38	0.26	0.25	0.59
(960,120)	0.71	7.55	3.26	0.64	0.71	0.63	0.71	0.14	0.25	0.53
(μ_1^{-1}, μ_2^{-1})										
(5,20)	0.04	10.81	2.94	0.04	0.04	0.04	0.04	0.14	0.09	0.35
(5,25)	0.01	7.81	2.44	0.01	0.01	0.01	0.01	0.13	0.03	0.19
(5,30)	0.00	9.50	1.85	0.01	0.00	0.01	0.00	0.10	0.01	0.10
(10,20)	0.63	8.41	4.50	0.58	0.63	0.56	0.63	0.39	0.33	0.52
(10,25)	0.20	8.82	5.56	0.19	0.20	0.18	0.20	0.40	0.23	0.42
(10,30)	0.07	11.54	5.60	0.08	0.07	0.08	0.07	0.54	0.15	0.32
(15,20)	1.78	4.91	3.63	1.55	1.78	1.51	1.80	0.16	0.40	0.55
(15,25)	0.71	8.25	5.42	0.66	0.71	0.62	0.71	0.35	0.30	0.45
(15,30)	0.28	11.42	6.47	0.31	0.28	0.27	0.28	0.55	0.24	0.38
$(R_1(0), R_2(0))$										
(0.98,0.8)	0.60	7.41	4.37	0.52	0.60	0.51	0.60	0.43	0.40	0.23
(0.98,0.5)	0.06	11.81	4.06	0.13	0.06	0.11	0.06	0.18	0.32	0.13
(0.9,0.8)	0.74	6.92	4.34	0.62	0.74	0.61	0.74	0.38	0.40	0.26
(0.9,0.5)	0.08	11.67	4.27	0.12	0.08	0.10	0.08	0.09	0.28	0.15

References

Almogy, G., Belzberg, H., Mintz, Y., Pikarsky, A.K., Zamir, G., and Rivkind, A.I. (2004). Suicide bombing attacks: update and modifications to the protocol. *Annals of Surgery, 239(3),* 295-303.

Almogy, G., Mintz, Y., Zamir, G., Bdolah-Abram, T., Elazary, R., Dotan, L., Faruga, M., and Rivkind, A.I. (2006). Suicide bombing attacks: can external signs predict internal injuries? *Annals of Surgery, 243(4),* 541-546.

Argon, N.T., Ziya, S., and Righter, R. (2008). Scheduling impatient jobs in a clearing system with insights on patient triage in mass casualty incidents. *Probability in the Engineering and Informational Sciences, 22,* 301-332.

Armstrong, J.H., Frykberg, E.R., and Burris, D.G. (2008). Toward a national standard in primary mass casualty triage. *Disaster Medicine and Public Health Preparedness, 2(S1),* S8-S10.

Aschkenasy-Steuer, G., Shamir, M., Rivkind, A., Mosheiff, R., Shushan, Y., Rosenthal, G., Mintz, Y., Weissman, C., Sprung, C.L., and Weiss, Y.G. (2005). Clinical review: the Israeli experience: conventional terrorism and critical care. *Critical Care, 9(5),* 490-499.

Ashkenazi, I. Kessel, B., Khashan, T., Haspel, J., Oren, M., Olsha, O., and Alfici, R. (2006). Precision of in-hospital triage is mass-casualty incidents after terrorist attacks. *Prehospital and Disaster Medicine, 21(1),* 20-23.

Ashkenazi, I. Kessel, B., Olsha, O., Khashan, T., Oren, M., Haspel, J., and Alfici, R. (2008). Defining the problem, main objective, and strategies of medical management in mass-casualty incidents caused by terrorist events. *Prehospital and Disaster Medicine, 23(1),* 82-89.

Aylwin, C.J., König, T.C., Brennan, N.W., Shirley, P.J., Davies, G., Walsh, M.S., and Brohi, K. (2006). Reduction in critical mortality in urban mass casualty incidents: analysis of triage, surge, and resource use after the London bombings on July 7, 2005. *The Lancet, 368,* 2219-2225.

Beliën, J. and Demeulemeester E. (2007). Building cyclic master surgery schedules with leveled resulting bed occupancy. *European Journal of Operational Research, 176*, 1185-1204.

Beliën, J., Demeulemeester, E., and Cardoen, B. (2009). A decision support system for cyclic master surgery scheduling with multiple objectives. *Journal of Scheduling, 12(2)*, 147-161.

Blake, J. and Carter, M. (2002). A goal programming approach to strategic resource allocation in acute care hospitals. *European Journal of Operational Research, 140*, 541-561.

Blake, J. and Donald, J. (2002). Mount Sinai Hospital uses integer programming to allocate operating room time. *Interfaces, 32(2)*, 63-73.

Boots, N.K. and Tijms, H. (1999). A multiserver queueing system with impatient customers. *Management Science, 45(3)*, 444-448.

Brumelle, S. and Walczak, D. (2003). Dynamic airline revenue management with multiple semi–Markov demand. *Operations Research, 51*, 137–148.

Cardoen, B., Demeulemeester, E., and Beliën, J. (2009). Sequencing surgical cases in a day-care environment: an exact branch-and-price approach. *Computers & Operations Research, 36(9)*, 2660-2669.

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

Chow, V.S., Puterman, M.L., Salehirad, N., Huang, W., and Atkins, D. (2008). Reducing surgical ward congestion at the Vancouver Island Health Authority through improved surgical scheduling. Technical Report, Centre for Operations Excellence, The University of British Columbia.

Christie, P.M.J. and Levary, R.R. (1998). The use of simulation in planning the transportation of patients to hospitals following a disaster. *Journal of Medical Systems, 22(5)*, 289-299.

Cook, C.H., Muscarella, P., Praba, A.C., Melvin, W.S., and Martin, L.C. (2001). Reducing overtriage without compromising outcomes in trauma patients. *Archives of Surgery, 136*, 752-756.

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

Denton, B.T., Miller, A.J., Balasubramanian, H.J., and Huschka, T.R. (2010). Optimal allocation of surgery blocks to operating rooms under uncertainty. *Operations Research, 58(4)*, 802-816.

Dexter, F., Macario, A., and Traub, R.D. (1999a). Which algorithm for scheduling elective add-on cases maximizes operating room utilization? Use of bin packing algorithms and fuzzy constraints in operating room management. *Anesthesiology*, *91*, 1491-1500.

Dexter, F., Macario, A., Qian, F., and Traub, R.D. (1999b). Forecasting surgical groups' total hours of elective cases for allocation of block time: application of time series analysis to operating room management. *Anesthesiology, 91*, 1501-1508.

Dexter, F. and 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*, 933-942.

Dexter, F., Traub, R., and Macario, A. (2003). How to release allocated operating room time to increase efficiency: predicting which surgical service will have the most under-utilized operating room time. *Anesthesia & Analgesia*, *96*, 507-512.

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

Einav, S., Feigenberg, Z., Weissman, C., Zaichik, D., Caspi, G., Kotler, D., and Freund, H.R. (2004). Evacuation priorities in mass casualty terror-related events. *Annals of Surgery, 239(3)*, 304-310.

Frykberg, E.R. and Tepas J.J. (1988). Terrorist bombings: lessons learned from Belfast to Beirut. *Annals of Surgery, 208(5)*, 569-576.

Frykberg, E.R. (2002). Medical management of disasters and mass casualties from terrorist bombings: how can we cope? *Journal of Trauma, 53(2)*, 201-212.

Frykberg, E.R. (2004). Principles of mass casualty management following terrorist disasters. *Annals of Surgery, 239(3)*, 319-321.

Frykberg, E.R. (2005). Triage: principles and practice. *Scandinavian Journal of Surgery, 94*, 272-278.

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

Glazebrook, K.D., Ansell, P.S., Dunn, R.T., and Lumley, R.R. (2004). On the optimal allocation of service to impatient tasks. *Journal of Applied Probability, 41*, 51-72.

Global Terrorism Database (2010). National Consortium for the Study of Terrorism and Responses to Terrorism (START). http://www.start.umd.edu/gtd/. Accessed on March 23, 2011.

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

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

Halpern, P., Tsai, M.-C., Arnold, J.L., Stok, E., and Ersoy, G. (2003). Mass-casualty, terrorist bombings: implications for emergency department and hospital emergency response (part II). *Prehospital and Disaster Medicine, 18(3)*, 235-241.

Hans, E., Wullink, G., van Houdenhoven, M., and Kazemier, G. (2008). Robust surgery loading. *European Journal of Operational Research, 185*, 1038-1050.

Hirshberg, A., Stein, M., and Walden, R. (1999). Surgical resource utilization in urban terrorist bombing: a computer simulation. *Journal of Trauma, 47(3)*, 545-550.

Hirshberg, A., Holcomb, J.B., and Mattox, K.L. (2001). Hospital trauma care in multiple-casualty incidents: a critical view. *Annals of Emergency Medicine, 37(6)*, 647-652.

Hirshberg, A. (2004). Multiple casualty incidents: lessons from the front line. *Annals of Surgery, 239(3)*, 322-324.

Hirshberg, A., Scott, B.G., Granchi, T., Wall, M.J., Mattox, K.L., and Stein, M. (2005). How does casualty load affect trauma care in urban bombing incidents? A quantitative analysis. *Journal of Trauma, 58(4)*, 686-695.

Hupert, N. Hollingsworth E., and Xiong, W. (2007). Is overtriage associated with increases mortality? Insights from a simulation model of mass casualty trauma care. *Disaster Medicine and Public Health Preparedness, 1 (1 Suppl)*, S14-S24.

Iravani, F. and Balcıoğlu, B. (2008). On priority queues with impatient customers. *Queueing Systems, 58*, 239-260.

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

Jenkins, J.L., McCarthy, M.L., Sauer, L.M., Green, G.B., Stuart, S., Thomas, T.L., and Hsu, E.B. (2008). Mass-casualty triage: time for an evidence-based approach. *Prehospital and Disaster Medicine, 23(1)*, 3-8.

Kluger, Y. (2003). Bomb explosions in acts of terrorism – detonation, wound ballistics, triage, and medical concerns. *Israel Medical Association Journal, 5*, 235-240.

Kluger, Y., Peleg, K., Daniel-Aharonson, L., and Mayo, A. (2004). The special injury pattern in terrorist bombings. *Journal of the American College of Surgeons, 199(6)*, 875-879.

Kosashvili, Y., Aharonson-Daniel, L., Peleg, K., Horowitz, A., Laor, D., and Blumenfeld, A. (2009). Israeli hospital preparedness for terrorism-related multiple casualty incidents: can the surge capacity and injury severity distribution be better predicted? *Injury, 40(7)*, 727-731.

Lamiri, M., Xie, X., and Zhang, S. (2008a). Column generation approach to operating theater planning with elective and emergency patients. *IIE Transactions, 40*, 838-852.

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

Lee, T.C. and Hersh, M. (1993). A model for dynamic airline seat inventory control with multiple seat bookings. *Transportation Science, 27(3)*, 252–265.

Lerner, E.B., and Moscati, R.M. 2001. The golden hour: scientific fact or medical "urban legend"? *Academic Emergency Medicine, 8(7)*, 758-760.

Lerner, E.B., Schwartz, R.B., Coule, P.L, Weinstein, E.S., Cone, D.C, Hunt, R.C., Sasser, S.M., Liu, J.M., Nudell, N.G., Wedmore, I.S., Hammond, J., Bulger, E.M., Salomone, J.P., Sanddal, T.L., Lord, G.C., Markenson, D., and O'Connor, R.E. (2008). Mass casualty triage: an evaluation of the data and development of a proposed national guideline. *Disaster Medicine and Public Health Preparedness, 2(Suppl 1)*, S25-S34.

Li, D. and Glazebrook, K.D. (2010). An approximate dynamic programming approach to the development of heuristics for the scheduling of impatient jobs in a clearing system. *Naval Research Logistics, 57*, 225-236.

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

McManus, M., Long, M., Cooper, A., Mandell, J., Berwick, D., Pagano, M., and Litvak, E. (2003). Variability in surgical caseload and access to intensive care services. *Anesthesiology. 98(6)*, 1491-1496.

Meredith, W., Rutledge, R., and Hansen, A.R. (1995). Field triage of trauma patients based on the ability to follow commands: a study in 29,573 injured patients. *Journal of Trauma, 38*, 129-135.

Min, D. and Yih, Y. (2010). Scheduling elective surgery under uncertainty and downstream capacity constraints. *European Journal of Operational Research, 2006*, 642-652.

Ogulata, S.N. and Erol, R. (2003). A hierarchical multiple criteria mathematical programming approach for scheduling general surgery operations in large hospitals. *Journal of Medical Systems, 27(3)*, 259-270.

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

Peleg, K. and Kellerman, A.L. (2009). Enhancing hospital surge capacity for mass casualty events. *Journal of the American Medical Association, 302(5)*, 565-567.

Pham, D., and Klinkert, A. (2008). Surgical case scheduling as a generalized job shop scheduling problem. *European Journal of Operational Research*. *185*, 1011–1025.

Powell, W.B. (2007). *Approximate Dynamic Programming: Solving the Curse of Dimensionality*. Wiley. Hoboken, NJ.

Price, C., Golden, B., Harrington, M., Konewko, R., Wasil, E., and Herring, W. (2011). Reducing boarding in a post-anesthesia care unit. *Production and Operations Management*, forthcoming.

Rothman, R.E., Hsu, E.B., Kahn, C.A., and Kelen, G.D. (2006). Research priorities for surge capacity. *Academic Emergency Medicine*, *13*, 1160-1168.

Sacco, W.J., Navin, D.M., Fiedler, K.E., Waddell, R.K., Long, W.B., and Buckman, R.F. (2005). Precise formulation and evidence-based application of resource-constrained triage. *Academic Emergency Medicine, 12*, 759-770.

Samanloiglu, F., Ayag, Z., Batili, B., Evcimen, E., Yilmaz, G., and Atalay, O. (2010). Determining master schedule of surgical operations by integer programming: a case study. *Proceeding of the 2010 Industrial Engineering Research Conference*, June 5-9, Cancun, Mexico.

Santibañez, P., Begen, M., and 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*, 269-282.

Schütz , H.-J. and Kolisch, R. (2010a). Approximate dynamic programming for capacity allocation in the service industry. *Working paper, available at SSRN: http://ssrn.com/abstract=1618315*.

Schütz , H.-J. and Kolisch, R. (2010b). Capacity allocation for demand of different customer–product–combinations with cancellation, no–shows, and overbooking when there is a sequential delivery of service. *Working paper, available at SSRN: http://ssrn.com/abstract=1618313*.

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

Strum, D.P. Vargas, L.G., and May, J.H. (1999). Surgical subspecialty block utilization and capacity planning: a minimal cost analysis model. *Anesthesiology, 90(4)*, 1176-1185.

Subramanian, J., Stidham Jr., S., and Lautenbacher, C.J. (1999). Airline yield management with overbooking, cancellations, and no-shows. *Transportation Science*, *33(2)*, 147–167.

Tanfani, E. and Testi, A. (2010). A pre-assignment heuristic algorithm for the master surgical schedule problem (MSSP). *Annals of Operations Research, 178*, 105–119.

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

Tijms, H.C. (1994). *Stochastic Models: An Algorithmic Approach*. Wiley. Chichester, UK.

Turégano-Fuentes, F., Pérez-Diaz, D., Sanz-Sánchez, M., and Alonso, J.O. (2008). Overall assessment of the response to the terrorist bombings in trains, Madrid, 11 March 2004. *European Journal of Trauma and Emergency Surgery, 5*. 433-441.

Valdez, R.S., Ramly, E., and Brennan, P.F. (2010). Industrial and Systems Engineering and Health Care: Critical Areas of Research--Final Report. (Prepared by Professional and Scientific Associates under Contract No. 290-09-00027U.) AHRQ Publication No. 10-0079. Rockville, MD: Agency for Healthcare Research and Quality.

Van Mieghem, J. (1995). Dynamic scheduling with convex delay costs: the generalized cμ rule. *Annals of Applied Probability*, *5(3)*, 808-833.

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

Zhang, B., Murali, P., Dessouky, M.M., and Belson, D. (2009). A mixed integer programming approach for allocating operating room capacity. *Journal of the Operational Research Society, 60*, 663-673.

Zhao, Z.-X., Panwar, S.S., and Towsley, D. (1991). Queueing performance with impatient customers. *Proceedings of IEEE INFOCOM'91, 1*, 400-409.