

An approximate dynamic programming approach to the admission control of elective patients

Jian Zhang^{a,b,*}, Mahjoub Dridi^b, Abdellah El Moudni^b

^a School of Industrial Engineering, Eindhoven University of Technology, 5600MB Eindhoven, Netherlands

^b Laboratoire Nanomédecine, Imagerie et Thérapeutiques, Université Bourgogne Franche-Comté, UTBM, Rue Thierry Mieg, 90010 Belfort cedex, France

* Corresponding author

Email addresses: j.zhang4@tue.nl (J. Zhang)
mahjoub.dridi@utbm.fr (M. Dridi)
abdellah.el-moudni@utbm.fr (A. El Moudni)

ABSTRACT

In this paper, we propose an approximate dynamic programming algorithm to solve a Markov decision process (MDP) formulation for the admission control of elective patients. To manage elective patients from multiple specialties equitably and efficiently, we establish a waiting list of patients and assign each patient a time-dependent dynamic priority score. Then taking the random arrivals of patients into account, sequential decisions are made on a weekly basis. At the end of each week, we select the patients to be treated in the following week from the waiting list. By minimizing the cost function of MDP over an infinite horizon, we seek to achieve the best trade-off between the patients' waiting time and the over-utilization of operating rooms and downstream resources. Considering the curses of dimensionality resulting from the large scale of realistically sized problems, we first analyze the structural properties of the MDP model and propose an algorithm that facilitates the search for greedy actions. We then develop a novel approximate dynamic programming algorithm based on recursive least-squares temporal difference learning as the solution technique. Experimental results reveal that the proposed algorithms consume much less computation time in comparison with that required by conventional dynamic programming methods. Additionally, the algorithms are shown to be capable of computing high-quality near-optimal policies for realistically sized problems.

Keywords: patient admission control, operating theatre planning, Markov decision process, approximate dynamic programming, recursive least-squares temporal difference learning

1 Introduction

Globally, the aging population and a rising quality of life are driving the demand for health services to increase rapidly, leading to shortages of medical resources and imposing a heavy financial burden on national governments. In the United States, the healthcare expenditure has been increasing continually, reaching \$3.5 trillion in 2017 and accounting for 17.9% of the gross domestic product (Centers for Medicare & Medicaid Services, 2018). The same figures for Australia in 2015–2016 were \$170 billion and 10.3%, respectively, while health expenditure increased (by 50%) much faster than the population growth (17%) (Australian Institute of Health and Welfare, 2018). During the period from 2010 to 2017, China increased its government spending on health and the number of medical personnel employed by 163% and 43%, respectively. However, its health expenditure per capita was still much lower than that of developed countries and the increase in its health service capacity was not enough to cope with the rising demands on the service (Xiao et al., 2016; National Bureau of Statistics of China, 2019). The same challenges have also been faced by European countries. For example, Portugal's surgical demand grew by 43.7% from 2006 to 2014, while the median waiting time for surgery reached 3.0 months and 12% of patients waited longer than their clinically recommended maximum waiting time (Marques & Captivo, 2017).

In order to satisfy a growing demand for health services and to slow down the increase in health expenditure, hospital managers should improve the efficiency and quality of healthcare activities while reducing the hospitals' expenditures as much as possible. In a hospital, the operating theatre (OT, consisting of operating rooms (ORs) and downstream facilities) is generally considered to be both the main revenue center as well as the most expensive department since providing surgical services consumes more than 40% of the hospital's budget and contributes a similarly large proportion to the hospital's total revenues (Denton et al., 2007). Therefore, the management of OT and the scheduling of surgeries have drawn much attention from both researchers and practitioners.

The complexity of OT management problems is the result of many factors including the expensiveness and shortage of OT resources (Rath et al., 2017), the conflicting interests of different stakeholders (e.g., patients and

hospitals) (Marques & Captivo, 2017), and the uncertainty associated with surgical activities (Banditori et al., 2013; Jebali & Diabat, 2015). For these reasons, researchers have developed and applied various operations research methodologies to cope with OT management problems. In the various relevant works, OT management decisions are generally divided into three hierarchical levels (Guerriero & Guido, 2011; Zhu et al., 2019): the strategic level determines the distribution of OR capacity among different specialties on a long-term basis; the tactical level involves the development of a master surgery schedule (MSS) for one or several months; the operational level concerns the assignment of a definite date and a specific OR for each elective surgery (advance scheduling) and the intra-day sequencing of the scheduled surgeries (allocation scheduling). The three decision levels are interrelated since each level depends on the decisions made at the higher level (Koppka et al., 2018).

This paper addresses the admission control of elective patients at the operational level. It is assumed that the allocation of OT resources among specialties is already determined by the strategic level and that an MSS has been fixed at the tactical level. A dynamic waiting list is built to manage the elective patients according to their specialties, urgency coefficients and actual waiting times. At the end of each week, the surgery planner updates the waiting list by adding the newly arrived patients and removing the postoperative ones, then determines the number and type of elective cases that will be performed in the next week. Our objective is to shorten patients' waiting times and to optimize the utilization of OT resources (ORs and downstream recovery beds). Emergency cases are not involved in the problem studied in this paper because they can be treated by dedicated facilities (Dios et al., 2015; Guido & Conforti, 2017). Since uncertainty is a key feature of surgical activities and significantly impacts the efficiency and quality of surgical services (Guerriero & Guido, 2011; Jebali & Diabat, 2015), three major sources of uncertainty are considered in our mathematical model and/or our numerical experiments: new patient arrivals, surgery durations and the length of stay (LOS) of postoperative patients in downstream facilities.

The patient admission control problem studied in this paper can be regarded as a subproblem of advance scheduling. In a typical advance scheduling problem, the surgery planner determines the patients to be treated in the current planning period (usually one week) and assigns these patients to specific ORs and surgery dates (e.g., Min & Yih, 2010b; Jebali & Diabat, 2015; Neyshabouri & Berg, 2017; Marques & Captivo, 2017; Moosavi & Ebrahimnejad, 2018). Such a problem is usually formulated as a pure mathematical programming model that only optimizes the cost of one single planning period without considering the impact on subsequent periods. However, two consecutive planning periods are always correlated since the postponed surgeries of the present period will continue to incur waiting costs or surgery costs in the next period. Therefore, optimizing the costs of each period separately cannot guarantee global optimality. In contrast, a patient admission control problem is usually formulated as a Markov decision process (MDP) (e.g., Patrick et al., 2008; Min & Yih, 2010a, 2014; Astaraky & Patrick, 2015; Truong, 2015). It focuses on the management of the patient waiting list and determines the selection of patients to be treated in each planning period (usually one week or one day). Though the MDP model does not address intra-week or intra-day assignment decisions, it optimizes the flow of patients and minimizes the expected total costs over an infinite horizon. Hence, the optimal policy of an MDP offers a better long-term performance than the myopic policy provided by a pure mathematical programming model.

In this paper, we propose an MDP model for patient admission control that considers both the needs of multiple specialties and the limited capacity of ORs and downstream facilities (recovery beds in the surgical intensive care unit (SICU)), as well as time-dependent dynamic patient priority scores. Sequential decisions are made at the end of each week that determine the patients to be treated in the following week. Since the MDP model does not capture intra-week scheduling, hospital-related costs (incurred by overusing ORs and the SICU) cannot be exactly computed. Hence, for a given selection of patients, the cost function of the MDP model computes the exact patient-related costs incurred by performing and postponing surgeries and estimates the hospital-related costs based on total resource

availability and the scheduled patients’ total expected surgery durations and LOSs. While taking into account the random arrivals of new patients in each week, the MDP model provides an optimal policy that leads to the lowest expected total costs over an infinite horizon. Once the selection of patients is determined, these patients’ assignments to surgical blocks (made up of a combination of ORs and dates) can be optimized using a stochastic programming model. For example, J. Zhang et al. (2019b) propose a two-level optimization model that combines MDP and stochastic programming to optimize all the decisions required by an advance scheduling problem. Their model addresses simple instances with a single specialty and two surgical blocks, while the MDP model proposed in this paper can be combined with a more complicated stochastic programming model in future research and can thereby be used to solve more realistic advance scheduling problems.

Compared to existing research into patient admission control (Patrick et al., 2008; Min & Yih, 2010a, 2014; Astaraky & Patrick, 2015; Truong, 2015), this work presents two main innovations. First, it proposes a comprehensive patient admission control model that simultaneously incorporates time-dependent dynamic patient priority scores, the needs of multiple specialties, and the capacity constraints of ORs and SICU. Second, in order to tackle the curses of dimensionality for realistically sized problems, this paper investigates the mathematical properties of the proposed model in depth and develops a novel approximate dynamic programming (ADP) algorithm based on recursive least-squares temporal difference (RLS-TD(λ)) learning. The proposed algorithm is tested through extensive numerical experiments and its capability to solve realistically sized problems is validated.

The remainder of this paper is organized as follows: Section 2 provides a comprehensive review of the existing literature on patient admission control and MDP. Section 3 describes the studied patient admission control problem and presents the infinite-horizon MDP formulation. Section 4 addresses the structural analysis for the MDP model and proposes an algorithm to predigest the exploration of action space. The RLS-TD(λ)-based ADP algorithm is introduced in Section 5, and the results of numerical experiments are presented and discussed in Section 6. Finally, the conclusions and several possible future extensions of this work are given by Section 7.

2 Literature Review

In this section, we review the existing literature relevant to patient admission control and the MDP.

2.1 Patient admission control

Typically, in a patient admission control problem, the surgery planner manages a dynamic waiting list of elective patients and makes sequential decisions at the beginning (or end) of each planning period (day or week) to determine which patients are treated in the present (or the next) period. Since the patient-to-block assignments are not considered, patient admission control can be regarded as a subproblem of advance surgery scheduling. Among the existing research on advance surgery scheduling, mathematical programming (MP) models are commonly used to optimize the cost of one single planning period (e.g., Lamiri, Xie, Dolgui, & Grimaud, 2008; Min & Yih, 2010b; Jebali & Diabat, 2015; Neyshabouri & Berg, 2017; Moosavi & Ebrahimnejad, 2018). However, these MP models minimize the short-term cost myopically and may lead to high costs-to-go in the future. In contrast, patient admission control is usually formulated as an MDP model in order to minimize the expected total costs over a long planning horizon. Though the MDP model does not address the issue of patient-to-block assignments, it can be combined with an MP model to optimize all the advance scheduling decisions. Such a combination may lead to a much better long-term performance in comparison to that achieved by a pure MP model (J. Zhang et al., 2019b). This paper aims to formulate and efficiently solve a comprehensive MDP model for realistic patient admission control problems, while the optimization of patient-to-block assignments is left for further research.

In the literature on patient admission control and surgery scheduling, elective patients and non-elective patients are two commonly addressed patient groups. The former group is usually added onto a waiting list before being treated. These patients' surgeries can be postponed for some period of time. In comparison, the latter group is made up of emergent patients (emergencies) that should be treated as soon as possible. Among the relevant research, two policies are commonly adopted to deal with the two different patient groups. The first one is a dedicated policy under which all emergent patients are channeled to dedicated surgical facilities, so that scheduled elective surgeries are not interrupted. When this policy is employed, the surgery planner focuses on the planning of elective surgeries using non-dedicated resources, while emergent patients can usually be ignored since they are treated in dedicated facilities on a first-come-first-served (FCFS) basis (e.g., Fei, Chu, & Meskens, 2009; Denton et al., 2010; Jebali & Diabat, 2015; Addis et al., 2016; Marques & Captivo, 2017; Neyshabouri & Berg, 2017; Roshanaei et al., 2017b, 2020; Y. Zhang et al., 2020). The second policy is more flexible. It allows emergent surgeries to be performed in any unoccupied OR. Scheduling surgeries under this flexible policy is more complex than doing so under the dedicated policy since emergencies introduce more uncertainty into the scheduling problem. In research into the employment of the flexible policy, the authors usually assume the emergency demand to be a stochastic parameter and optimize the surgical resources (e.g., OR capacity and recovery beds) reserved for emergencies in a way that balances the satisfaction of elective patients and the quick access to care of emergent ones (e.g., Y. Wang et al., 2014; Truong, 2015; Rachuba & Werners, 2017). In addition to the two policies discussed above, some papers plan elective and non-elective surgeries using a hybrid policy that maintains both dedicated and versatile ORs (e.g., Tancrez et al., 2013; Hosseini & Taaffe, 2014; Ferrand et al., 2014). Ferrand et al. (2010) present a study of a two-OR planning problem. They conclude that adopting the dedicated policy reduces the elective patients' waiting times and the overtime of ORs, but that it leads to longer waiting times for emergent patients. Duma and Aringhieri (2019) provide a detailed comparison that reveals that the dedicated policy results in fewer elective surgery cancellations, higher resource utilization rates, and shorter waiting lists, and that the flexible policy brings about a better trade-off between the interests of elective and non-elective patients. Moreover, they establish that a further improvement can be achieved through a mixture of the two policies (i.e., hybrid policies). The authors also claim that the performances of different policies depend on the scenario and the operative conditions at hand. Hence, any policy could be the best one under a specific problem setting. **In this paper, we assume a dedicated policy meaning that emergencies are excluded from our patient admission control problem.**

Effectively managing the waiting list of elective patients is the key to patient admission control. To determine the relative priority of elective patients on the waiting list, it is important to adopt a proper prioritization system (Guerriero & Guido, 2011; Van Riet & Demeulemeester, 2015). Patrick et al. (2008) classify elective patients into several urgency-related groups (URGs) with static priority scores and specify a priority-related waiting-time target for each group. However, Testi et al. (2007) and Testi et al. (2008) suggest that classifying patients into URGs is not enough to guarantee the efficiency and equity of schedules. They propose a time-dependent prioritization scoring system that uses the product of urgency coefficient and waiting time to capture a patient's priority. Further to this, Valente et al. (2009) introduce a pre-admission model that has been put into practice in an Italian hospital. This model assesses the urgency level of each new patient and thereby determines a corresponding maximum time before treatment (MTBT). It then prioritizes the patient in real-time according to his/her urgency level and actual waiting time. Hospital data show that this pre-admission model allows homogeneous and standardized prioritization and enhances transparency, efficiency and equity. Min and Yih (2014) conduct numerical simulations that compare the effects of static and time-dependent dynamic prioritization on patients' waiting time. The results indicate that the waiting time for non-urgent patients is much longer than that of urgent ones when the static priority score is applied, whereas the dynamic priority score minimizes the total weighted waiting time of all patients, so that

the gap in the length of waiting time between urgent patients and non-urgent ones is significantly reduced. In the relevant research on advance surgery scheduling and patient admission control, time-dependent priority scores are widely used as multipliers of the patient-related costs (including surgery costs and waiting costs, e.g., Addis et al., 2016; Agnetis et al., 2012, 2014; Aringhieri et al., 2015; Jebali & Diabat, 2015, 2017; Lamiri, Xie, & Zhang, 2008; Lamiri, Xie, Dolgui, & Grimaud, 2008; Marques & Captivo, 2017; Min & Yih, 2010b; Rachuba & Werners, 2017; Roshanaei et al., 2017b, 2020; Tànfani & Testi, 2010; Testi & Tànfani, 2009; Vancroonenburg et al., 2019; S. Wang et al., 2016; J. Zhang et al., 2019a, 2019b, 2020). With this setting, patients with the longer waiting times are given the higher priority scores, and the surgery planner is encouraged to schedule the surgeries as early as possible because the unscheduled patients will incur higher costs in the future. In addition to the time-dependent priority scores, Neyshabouri and Berg (2017) introduce a new multiplier that addresses the relative importance of different specialties. This feature gives surgical cases from more important specialties (e.g., cardiology) higher priority than those from less important specialties (e.g., otolaryngology). Referring to these prioritization methods, we use the product of urgency coefficient, waiting time, and specialty-related importance factor to describe patient priority in this paper. Moreover, we assume that every patient is given a maximum allowed waiting time (dependent on his/her specialty and urgency coefficient) and that any patient that is not scheduled in the currently considered planning period incurs a waiting cost (multiplied by his/her time-dependent priority score, as mentioned above). Thus, the surgery planner gives priority to the patients who join the waiting list earlier.

While ORs are considered to be the most important surgical resources and their utilization is optimized in almost all the relevant works, recently, more and more researchers take the supporting facilities of surgical activities into consideration and address multi-resource patient admission control and surgery scheduling problems (e.g., Min & Yih, 2010b; Gocgun & Ghate, 2012; Huh et al., 2013; Astaraky & Patrick, 2015; Truong, 2015; Jebali & Diabat, 2015, 2017; Samudra et al., 2016; Neyshabouri & Berg, 2017; J. Zhang et al., 2019b, 2020). Among the various supporting facilities, the downstream recovery units, such as SICU, have drawn much attention from the researchers and practitioners, because the unavailability of SICU beds may block the postoperative patients in ORs and affect the feasibility of the ongoing surgery schedule (Min & Yih, 2010b; Jebali & Diabat, 2015). Moreover, Jonnalagadda et al. (2005) point out that the unavailability of recovery room is the cause of 15% of surgery cancellations in the hospital they have studied, and Utzolino et al. (2010) report that the readmission rate to SICU of the patients that are discharged due to the lack of recovery beds is almost 3 times that of the patients discharged electively. Given that SICU is an important surgical facility and usually becomes the bottleneck resource of an OT, in this work, we believe that considering the capacities of ORs and SICU together results in better resource utilization than only considering the OR capacity. By jointly planning ORs and SICU, the inconvenience and extra costs caused by exceeding the regular capacity of any of the two resources can be minimized, meanwhile the success rate of the intra-week surgery scheduling can be improved.

With different problem settings, patient admission control problems have been studied by many researchers. Gerchak et al. (1996) address a patient admission control problem in which decisions are made at the beginning of each day that determine the optimal number of patients to be treated that day. Similarly, Green et al. (2006) present a finite-horizon MDP formulation that deals with an intra-day diagnostic facility scheduling problem for multiple classes of patients. Further to this, Patrick et al. (2008) use an infinite-horizon MDP model to dynamically assign patients to diagnostic facilities. Their model differs from that of Gerchak et al. (1996) in that it considers multiple priority classes and quantifies the actual waiting times of elective patients. They also introduce a booking horizon that consists of a number of future days, so that patients can be assigned to anyone of the days within the booking horizon at each decision epoch. A similar booking horizon is adopted by Liu et al. (2010) to schedule single-priority outpatient appointments. Min and Yih (2010a) extend the model developed by Gerchak et al. (1996) to apply to

a multi-priority case; they then evaluate the impact of patient priority and cost settings on the resulting surgery schedules. Min and Yih (2014) go one step further and consider dynamic patient priority: each patient is initially assigned a fixed urgency coefficient when he/she joins the waiting list, then the product of the urgency coefficient and actual waiting time, which dynamically increases over time, determines his/her priority score. Astaraky and Patrick (2015) and Truong (2015) extend the model developed by Patrick et al. (2008) to incorporate multiple surgical resources (ORs, recovery beds, etc.); however, the multi-priority setting is dropped in Truong (2015).

In this paper, we extend the aforementioned extant research by taking into account time-dependent dynamic patient priority scores, the capacity constraints of ORs and the SICU, and the needs of multiple specialties and the different patient characteristics with which they are presented. To the best of our knowledge, this is the first time that these factors are collated into a comprehensive MDP model in order to handle realistically large-sized patient admission control problems.

2.2 Markov decision process

MDP is a discrete mathematical model used to facilitate sequential decision-making in the presence of uncertainty (White, 2001). It is the most commonly used mathematical model by existing research into patient admission control (e.g., Patrick et al., 2008; Min & Yih, 2010a, 2014; Astaraky & Patrick, 2015; Truong, 2015). In an MDP model, the status of the problem under consideration is described by the state variables from a set named state space. The possible actions that can be taken are included in another set called the action space. At every discrete time point (i.e., decision epoch), the agent observes the state of the problem and selects an action from the action space. The current state and the selected action determine the instant cost (or revenue) of the problem and the probability distribution of the subsequent state to which the problem transfers at the next decision epoch (Puterman, 1994). The objective of an MDP model is to find the optimal policy that specifies a mapping from the state space to the action space and optimizes the value function (i.e., minimizes the expected total costs or maximizes the expected total revenues over all decision epochs) (Mausam & Kolobov, 2012).

Most algorithms used to solve an MDP are based on dynamic programming (DP), such as policy iteration (PI), value iteration (VI), real-time dynamic programming (RTDP) and its variants. VI and PI evaluate the entire state space iteratively, updating the value function until the improvement is lower than a given threshold. These algorithms employ brute-force search strategies and the computational resources (memory and time) they need are exponential to the problem size. As a result, solving realistically sized MDP models using VI or PI is usually intractable. In order to improve the computational efficiency, Barto et al. (1995) propose a RTDP algorithm that only evaluates a subset of the state space. At every decision epoch, RTDP explores the state space along several sampled trajectories that are rooted at the state that describes the problem’s current status (i.e., the initial state). Specifically, at each visited state (including the initial state), RTDP updates the value function and policy then randomly samples the next state to visit according to the probability distribution determined by the current state and the corresponding action under the current policy. When a certain number of states are evaluated, RTDP returns to the initial state and repeats the above procedure. Since RTDP only visits those states that are reachable from the initial state, it outperforms PI and VI in terms of computational efficiency. However, the original version of RTDP lacks a mechanism for convergence detection and a proper criterion for terminating the computation; hence, it may waste computational resources on those states where the value function has already converged, or terminate so early that the value function is still far from convergence. To overcome these drawbacks, several extensions of RTDP have been developed in the literature. Bonet and Geffner (2003) propose labelled-RTDP (LRTDP), which incorporates a labelling scheme into the original RTDP. They label the states where the value function no longer improves; in this way, the computational resources can be concentrated on the non-labelled states where the value function has not converged. McMahan et al. (2005)

and Smith and Simmons (2006) propose bounded-RTDP (BRTDP) and focused-RTDP (FRTDP), respectively. These extended RTDP algorithms compute both an upper bound and a lower bound on the optimal value function; the gap between the two bounds can then be used to judge whether the value function has converged and the exploration of the state space can be guided towards the poorly understood states (i.e., the states where the gap between the two bounds is large). Further to this, Sanner et al. (2009) improve BRTDP by introducing a value of perfect information (VPI) analysis to detect whether the policy at a state has converged. The VPI-RTDP algorithm they propose saves computational resources by not visiting the states where the value function has not converged but the policy has already converged.

Though the DP-based algorithms have advanced in different ways, their computational performance is highly dependent on the problem scale; i.e., the CPU time and memory consumed by the DP-based algorithms increase exponentially as the size of the MDP model increases. Specifically, J. Zhang et al. (2019a) employ VI and the above-mentioned RTDP algorithms to solve patient admission control problems. Their experimental results show that VPI-RTDP outperforms VI significantly and that it is the best DP-based algorithm in terms of accuracy and efficiency. However, the problem size that VPI-RTDP can cope with is still fairly limited since the CPU time consumed by VPI-RTDP increases greatly as the size of the MDP model becomes slightly larger. Therefore, more advanced algorithms should be developed that can solve realistically sized patient admission control problems efficiently.

In short, the challenges faced by DP-based algorithms are the so-called three curses of dimensionality: state space, action space, and outcome space (containing all the possible subsequent states of a state-action pair) (Powell, 2011). Regarding the large action spaces, many researchers addressing patient admission control perform structural analyses of their MDP models to reduce the number of actions that should be evaluated (e.g., Min & Yih, 2010a, 2014; Truong, 2015). Moreover, approximate dynamic programming (ADP) provides a set of powerful algorithms that are able to cope with large state spaces and outcome spaces. Firstly and the most importantly, since the dimension of the value function depends on the cardinality of the state space, the computational complexity caused by a large state space can be reduced by estimating the high-dimensional value function with a low-dimensional approximator. Secondly, the enumeration of a large outcome space can be avoided by randomly sampling a subsequent state to evaluate. To the best of our knowledge, the application of ADP in the literature on patient admission control is very limited, although we note that Patrick et al. (2008) and Astaraky and Patrick (2015) have developed a linear programming-based ADP algorithm and a simulation-based ADP algorithm, respectively.

In this paper, we formulate the patient admission control problem being studied as an MDP model with an infinite horizon. Recognizing that the problems involved in this paper are much larger than those addressed in the existing research (e.g., Patrick et al., 2008; Min & Yih, 2010a, 2014; Astaraky & Patrick, 2015; Truong, 2015), we reduce the computational complexity of solving the MDP model in two ways. First, we perform an in-depth analysis to investigate the optimal policy’s properties, based on which we develop an efficient algorithm to simplify the exploration of the action space and to accelerate the solution approaches employed. Second, we develop an ADP algorithm based on RLS-TD(λ), an efficient reinforcement learning method proposed by Xu et al. (2002). RLS-TD(λ) updates a low-dimensional linear function to estimate the high-dimensional value function; thus, the RLS-TD(λ)-based ADP algorithm is capable of providing near-optimal policies for large-scale patient admission control problems and consumes much less CPU time than do conventional DP-based algorithms. In the literature, a similar RLS-TD(λ)-based ADP algorithm has been proposed by Yin et al. (2017) to solve a real-time traffic signal control problem. Their experimental results show that the proposed ADP algorithm provides high-quality policies and is much more efficient than DP-based algorithms.

3 Problem description and formulation

This section describes the patient admission control problem in detail and presents the infinite-horizon MDP model. We manage ORs and SICU recovery beds in an OT that provides surgical service for elective patients from J specialties. Every specialty j has a relative importance factor denoted by v_j , and the patients in specialty j are divided into U_j urgency groups. A maximum allowed waiting time W_{ju} is assigned to specialty- j patients with urgency coefficient u , and every patient must be treated before his/her waiting time w exceeds W_{ju} . Surgery durations and LOSs are subject to uncertainty, whereas the regular capacities of the ORs and SICU are fixed. Extra use of an OR leads to a cost of c_o per hour, while a penalty of c_e per bed/per day is incurred if the recovery beds in SICU are insufficient for the actual demand (since some patients in need of intensive care have to be transferred to other recovery units with a lower level of care, or extra beds have to be added to SICU). It should be mentioned that the actual LOSs of some patients can be 0, which means that not all the patients need intensive care after surgery. In addition to the hospital-related costs, we consider two types of patient-related costs in this paper: 1) surgery costs incurred by the booked surgeries which require surgical staff's work, utilization of surgical facilities, etc.; 2) waiting costs incurred by the deferred surgeries which cause patient dissatisfaction and may lead to deterioration in patients' health. Let c_b and c_d be the unit surgery cost and the unit waiting cost, respectively; it is then reasonable to assume that $c_d > c_b$. This assumption is adopted in much of the relevant research (e.g., Min & Yih, 2010a; Jebali & Diabat, 2015, 2017; Neyshabouri & Berg, 2017), because, from a patient's perspective, it is always preferable for his/her surgery to be performed as early as possible, rather than it be postponed. Moreover, patient priority score is usually used as a multiplier of the patient-related costs in the literature (e.g., Lamiri, Xie, & Zhang, 2008; Min & Yih, 2010a, 2014; Marques & Captivo, 2017; Neyshabouri & Berg, 2017; Roshanaei et al., 2017b, 2020). Hence, considering the dynamic patient priority score $Pr = v_juw$, we assume that every scheduled patient produces a surgery cost $c_b v_j uw$, and that every unscheduled patient incurs a waiting cost $c_d v_j uw$. The two types of time-dependent patient-related costs are both incorporated into the cost function, so that patients' waiting times can be minimized.

From the existing research studying advance scheduling, we find that elective surgeries are usually planned over a fixed planning horizon (e.g., one week) (e.g., Fei et al., 2008; Fei, Meskens, et al., 2009; Fei, Chu, & Meskens, 2009; Fei et al., 2010; Denton et al., 2010; Min & Yih, 2010b; Y. Wang et al., 2014; Hashemi Doulabi et al., 2016; Marques & Captivo, 2017; Neyshabouri & Berg, 2017; Roshanaei et al., 2017a, 2017b; Moosavi & Ebrahimnejad, 2018; Pang et al., 2018). In such studies, decisions to assign the patients on the waiting list to specific ORs are made at the beginning (or the end) of each horizon, with the dates of such assignments being within the present (or the next) horizon. When new elective patients arrive, the surgery schedule of the present horizon has already been determined. As a result, they are left unscheduled until the beginning of the next horizon. The patient admission control problem studied in this paper is a subproblem of advance scheduling; therefore, we assume that the newly arrived patients cannot be scheduled during the week of their arrival (i.e., direct admission is not allowed). The same assumption is also adopted by other research that addresses patient admission control, such as Min and Yih (2010a, 2014) and J. Zhang et al. (2019b).

At the end of each week, we update the patient waiting list by removing the treated patients and adding the newly arrived ones. We then determine which patients will be treated during the week after the current week. Let n_{juw} be the total number of type- $\{juw\}$ patients (i.e., specialty- j patients with urgency coefficient u and actual waiting time w) on the waiting list; the state of the MDP model can then be defined as the vector of n_{juw} : $s = \{n_{juw} : j = 1, 2, \dots, J; u = 1, 2, \dots, U_j; w = 1, 2, \dots, W_{ju}\}$. Similarly, the action of the MDP model is defined as vector $a = \{m_{juw} : j = 1, 2, \dots, J; u = 1, 2, \dots, U_j; w = 1, 2, \dots, W_{ju}\}$, where m_{juw} denotes the number of scheduled

type- $\{juw\}$ patients. Let $A(s)$ be the set of feasible actions for s , then, for any $a \in A(s)$:

$$\begin{cases} m_{juw} \leq n_{juw} & \text{if } w < W_{ju} \\ m_{juw} = n_{juw} & \text{if } w = W_{ju} \end{cases} \quad (1)$$

Let \tilde{n}_{ju}^τ be the number of newly arrived specialty- j patients with urgency coefficient u during week τ ; the transition of state from week τ to week $\tau + 1$ can then be written as

$$\begin{cases} n_{ju,1}^{\tau+1} = \tilde{n}_{ju}^\tau & \forall j, u \\ n_{ju,w+1}^{\tau+1} = \underbrace{n_{juw}^\tau - m_{juw}^\tau}_{\text{red underline}} & \forall j, u, w < \underbrace{W_{ju}}_{\text{red circle}} \end{cases} \quad (2)$$

Then, we can obtain the transition probability function of the MDP model as follows:

$$p(s_\tau, a_\tau, s_{\tau+1}) = \prod_{j=1}^J \prod_{u=1}^{U_j} \left[p(\tilde{n}_{ju}^\tau = n_{ju,1}^{\tau+1}) \prod_{w=1}^{W_{ju}} p(n_{juw}^\tau - m_{juw}^\tau = n_{ju,w+1}^{\tau+1}) \right] \quad (3)$$

In (3), $p(\tilde{n}_{ju}^\tau = n_{ju,1}^{\tau+1}) \in (0, 1)$ is the probability that the number of newly arrived type- $\{ju, 0\}$ patients in week τ is equal to the total number of type- $\{ju, 1\}$ patients in state $s_{\tau+1}$. Its value is computed by the probability mass function of a Poisson distribution with an arrival rate \bar{n}_{ju} . $p(n_{juw}^\tau - m_{juw}^\tau = n_{ju,w+1}^{\tau+1}) \in \{0, 1\}$ is a deterministic term that indicates if the number of unscheduled type- $\{juw\}$ patients in week τ matches the total number of type- $\{ju, w+1\}$ patients in state $s_{\tau+1}$. Its value can either be 0 or 1.

In a patient admission control problem, we cannot compute the exact utilizations of surgical resources, which depend on the intra-week schedule (i.e., patient-to-OR assignments) and the realizations of stochastic parameters. It is straightforward to estimate the expected resource utilizations using the total capacities of ORs and SICU as well as the distributional information of stochastic parameters (e.g., Astaraky & Patrick, 2015; Gerchak et al., 1996; Gocgun & Ghate, 2012; Huh et al., 2013; Jebali & Diabat, 2017; Lamiri, Xie, Dolgui, & Grimaud, 2008; Min & Yih, 2014; Truong, 2015). Even so, for the problem studied in this paper, computing the exact expectations of OR overtime and SICU bed shortage is a considerable challenge, because this requires computing the expectations of $|J|+1$ truncated probability distributions (lognormal distributions are commonly adopted) for each state-action pair. In order to reduce the computational complexity, we approximate the expected OR overtime and SICU bed shortage by taking the sum of the expected surgery durations and LOSs of the scheduled patients, subtracting the available OR and SICU capacity, and taking the positive parts. This way of computing resource utilization is commonly used in the relevant research and has been justified (e.g., Astaraky & Patrick, 2015; Molina-Pariente, Fernandez-Viagas, & Framinan, 2015; Molina-Pariente, Hans, et al., 2015; S. Wang et al., 2016; Roshanaei et al., 2017a, 2017b, 2020). For example, Molina-Pariente, Fernandez-Viagas, and Framinan (2015) show that the stochasticity of surgery durations does not influence the average OR utilization; S. Wang et al. (2016) reveal that the under- and over-estimations of surgery durations are mostly counterbalanced, and that stochasticity has little influence on the quality of their deterministically optimized surgery schedules; Roshanaei et al. (2017a, 2020) anticipate that ignoring uncertainty does not drastically diminish the cost-savings achieved by their deterministic optimization approach. In the numerical experiments of this work, we compute the overtime of ORs and the excess of SICU capacity using large numbers of randomly sampled surgery durations and LOSs, thus the performance and robustness of our policies under uncertainty can be effectively evaluated.

Then, the cost function of the MDP model for state-action pair (s, a) is defined as

$$C(s, a) = c_b \sum_{j=1}^J \sum_{u=1}^{U_j} \sum_{w=1}^{W_{ju}} v_j u w m_{juw} + c_d \sum_{j=1}^J \sum_{u=1}^{U_j} \sum_{w=1}^{W_{ju}} v_j u w (n_{juw} - m_{juw}) \\ + c_o \sum_{j=1}^J \left(\sum_{u=1}^{U_j} \sum_{w=1}^{W_{ju}} m_{juw} \bar{d}_j - \rho_1 B_j \right)^+ + c_e \left(\sum_{j=1}^J \sum_{u=1}^{U_j} \sum_{w=1}^{W_{ju}} m_{juw} \bar{l}_j - \rho_2 R \right)^+ \quad (4)$$

where $(\cdot)^+ = \max\{\cdot, 0\}$; \bar{d}_j and \bar{l}_j are the expectations of surgery duration and LOS for specialty- j patients (we assume that patients from the same specialty have the same lognormal distributions for surgery duration and LOS); B_j is the regular OR capacity reserved for specialty j ; R is the regular capacity of SICU; $\rho_1, \rho_2 \in (0, 1]$ are the estimated availability rates of the ORs and recovery beds, respectively. ρ_1 and ρ_2 are incorporated into the cost function, since the intra-week scheduling cannot always fully utilize the surgical resources. **We assume that an MSS has been fixed at the tactical level, then the value of B_j can be computed according to the number of surgical blocks (i.e., OR-days) assigned to specialty j and the time length of each surgical block.**

It can be observed that **cost function (4) is independent of week τ** . Moreover, since the number and type of elective surgeries do not change too much over weeks (Guerriero & Guido, 2011), we can assume that the arrival rates \bar{n}_{ju} of new patients are constant. Thus, **the transition probability function (3) is also independent of τ** and the MDP model is **stationary** (Mausam & Kolobov, 2012). In order to capture more dynamics of the problem, one might want to shorten the time interval between two consecutive decision epochs from one week to one day. However, **making decisions on a daily basis requires us to consider the two following issues. First**, elective surgeries cannot be scheduled at weekends. As a result, making the same decision on different weekdays leads to different expected waiting time of patients and different costs-to-go in the future. **Second**, the regular OR capacity preallocated to each specialty by an MSS usually varies from Monday to Friday. Due to the two issues, the resource constraints and cost function are no longer independent of τ , and the state space of the non-stationary MDP model will be five times larger than that of the stationary one. Referring to the literature, we can find that the planning horizons of most **advance surgery scheduling problems are one week**. Therefore, as a subproblem of advance surgery scheduling, the patient admission control problem should be formulated and solved on a weekly basis.

Figure 1 demonstrates the evolution of the MDP model for an example patient admission control problem with two specialties ($j = 1, 2$) and two urgency groups ($u = 1, 2$) in each specialty.

Finally, the objective of the MDP model is to find the optimal policy π^* that minimizes the discounted expected costs over the infinite horizon:

$$\pi^* = \arg \min_{\pi} \mathbb{E} \left\{ \sum_{\tau=1}^{+\infty} \gamma^{\tau-1} C[s_{\tau}, \pi(s_{\tau})] \right\} \quad (5)$$

where $\gamma \in [0, 1)$ is the discount factor and $\pi(s_{\tau})$ is the action corresponding to s_{τ} under policy π .

4 Structural analysis

The MDP model presented in Section 3 can be regarded as a substantial extension of the MDP models formulated by **Gerchak et al. (1996) and Min and Yih (2010a, 2014)**. In these previous works, the authors explore the properties of their MDP models to facilitate the solution procedures. In this paper, as we take time-dependent patient priority scores, multiple resource constraints (ORs and SICU), and multiple specialties with different patient characteristics into consideration, our MDP model is significantly more complex and most of the model properties proved by Gerchak et al. (1996) and Min and Yih (2010a, 2014) do not apply to our MDP model. Hence, we investigate the structural

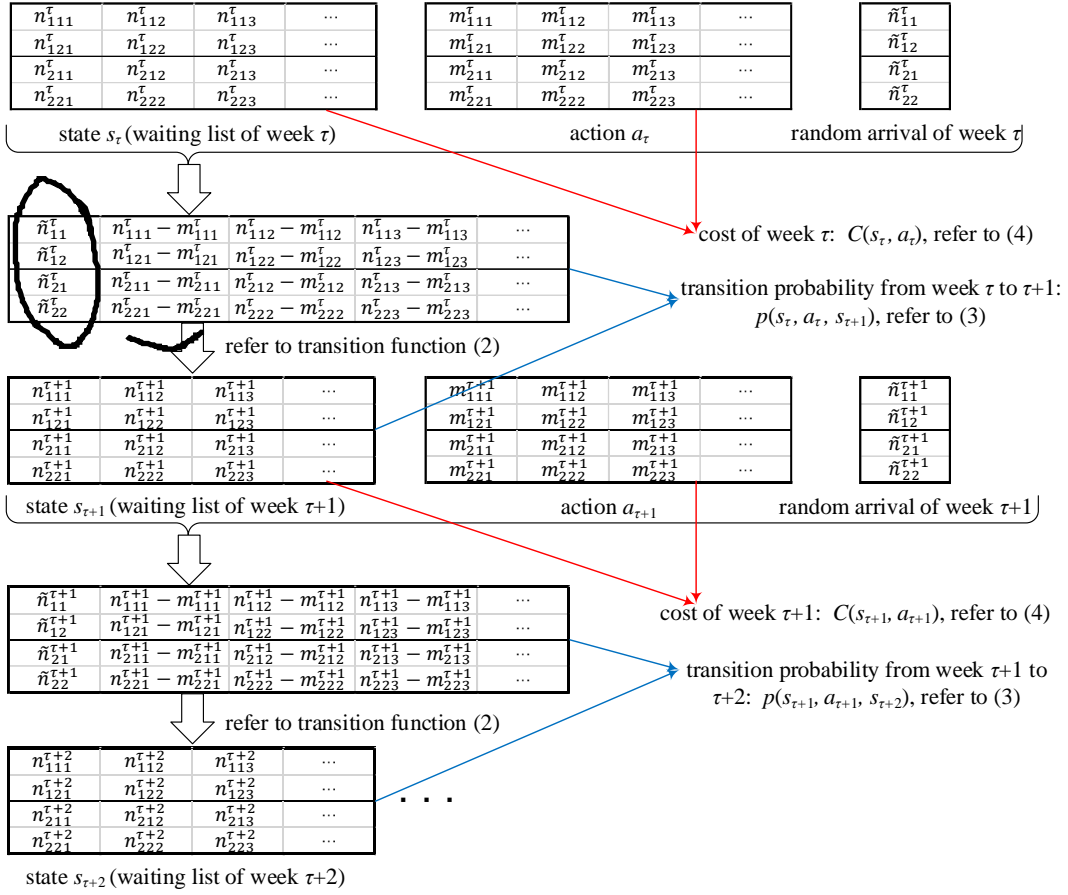


Figure 1: An example MDP model for a two-specialty, two-urgency-group problem

properties of our MDP model in this section, and these properties will help us to improve computational efficiency when evaluating the action space.

We first create a vector $\Delta_{j'u'w'} = \{\delta_{juw}\} \in S \cup A$ (S denotes the state space and A denotes the action space) in which $\delta_{j'u'w'} = 1$ and the other elements are 0. Then, two partial orders on $S \cup A$ are defined as follows:

Definition 1. Let $x = \{x_{juw}\} \in S \cup A$ and $y = \{y_{juw}\} \in S \cup A$.

- (i) If $\forall j, u, w: x_{juw} \leq y_{juw}$ and $\exists j', u', w'$ s.t. $x_{j'u'w'} < y_{j'u'w'}$, then $x < y$.
- (ii) If $\forall j, u, w: x_{juw} = y_{juw}$, then $x = y$.
- (iii) Otherwise, x and y are incomparable.

Definition 2. Let $x = \{x_{juw}\} \in S \cup A$ and $y = \{y_{juw}\} \in S \cup A$. The patient priority scores of x and y are denoted by $P(x)$ and $P(y)$, respectively.

- (i) If $x = y + \Delta_{j'u'w'} - \Delta_{j''u''w''}$ and $u'w' < u''w''$, then $P(x) < P(y)$.
- (ii) If $x = y$, then $P(x) = P(y)$.
- (iii) Otherwise, $P(x)$ and $P(y)$ are incomparable.

Next, we propose Lemma 1 that will be used in the proofs of structural properties.

Lemma 1. For functions $f, g : D \rightarrow \mathbb{R}$ with $D \subseteq \mathbb{R}$: $\min f(x) - \min g(x) \geq \min[f(x) - g(x)]$.

Proof. See Appendix A. \square

Let $C_0(s) = \min_{a \in A(s)} C(s, a)$ be the optimal single-period cost and $a_0(s) = \arg \min_{a \in A(s)} C(s, a)$ be the optimal single-period action. For the sake of simplicity, we use $C_p(s, a)$ to denote the patient-related cost, i.e., the sum of the first term and the second term of (4), and use $C_h(a)$ to denote the hospital-related cost, i.e., the sum of the third term and the fourth term of (4). Using Definition 1, 2 and Lemma 1, the following statements can be proved to be true:

- Proposition 1.** (i) $C_0(s)$ is increasing in s .
(ii) If $P(a)$ and $P(a')$ are comparable, then $C_h(a) = C_h(a')$ holds.
(iii) $C_0(s)$ is increasing in $P(s)$.

Proof. See Appendix B. \square

From (i) and (iii) of Proposition 1, we know that the optimal single-period cost $C_0(s)$ is monotonically increasing in n_{juw} and the priority scores of the patients on the waiting list. In addition, (ii) of Proposition 1 indicates that the expected hospital-related costs depend on the number of scheduled patients in each specialty and are independent of patients' urgency coefficients and waiting times. With Proposition 1, we can further analyze the optimal value function (i.e., the value function under the optimal policy π^*) which is given by

$$V^{\pi^*}(s) = \mathbb{E} \left\{ \sum_{n=1}^{+\infty} \gamma^{n-1} C[s_n, \pi^*(s_n)] \right\} = \sum_{n=1}^{+\infty} \gamma^{n-1} \mathbb{E}\{C[s_n, \pi^*(s_n)]\} \quad (6)$$

where $s_1 = s$. Since $C[s_n, \pi^*(s_n)]$ is finite and $\gamma < 1$, then $\gamma^{n-1} \mathbb{E}\{C[s_n, \pi^*(s_n)]\} \rightarrow 0$ as n goes to infinity. We can thereby assume $\gamma^{n-1} \mathbb{E}\{C[s_n, \pi^*(s_n)]\} = 0$ for any $n > N \gg 0$, then

$$V_n^{\pi^*}(s) = \begin{cases} C[s, \pi^*(s)] + \gamma \sum_{s' \in S} p[s, \pi^*(s), s'] V_{n+1}^{\pi^*}(s') & \text{if } n = 1, 2, \dots, N \\ 0 & \text{if } n = N + 1, N + 2, \dots \end{cases} \quad (7)$$

where $V_1^{\pi^*}(s) = V^{\pi^*}(s)$. Further, the Q-value of state-action pair (s, a) under π^* is defined as

$$Q^{\pi^*}(s, a) = C(s, a) + \gamma \sum_{s' \in S} p(s, a, s') V^{\pi^*}(s') \quad (8)$$

$Q^{\pi^*}(s, a)$ can also be written as the following recursive formula:

$$Q_n^{\pi^*}(s, a) = C(s, a) + \gamma \sum_{s' \in S} p(s, a, s') V_{n+1}^{\pi^*}(s') \quad (9)$$

where $Q_1^{\pi^*}(s, a) = Q^{\pi^*}(s, a)$ and $V_{n+1}^{\pi^*}(s') = \min_{a \in A(s')} Q_{n+1}^{\pi^*}(s', a)$.

Before exploring the properties of $V^{\pi^*}(s)$, we present the following notations and analysis, which will be useful when proving our propositions. Let vector $G_a^s = \{g_{juw}\}$ be the post-action state of (s, a) :

$$\begin{cases} g_{ju,1} = 0, & \forall j, u = 1, 2, \dots, U_j \\ g_{ju,w+1} = n_{juw} - m_{juw}, & \forall j, u = 1, 2, \dots, U_j, w = 1, 2, \dots, W_{ju} - 1 \end{cases} \quad (10)$$

and $\Psi = \{\psi_{juw}\}$ be the vector of newly arrived patients:

$$\begin{cases} \psi_{ju,1} = \tilde{n}_{ju}, & \forall j, u = 1, 2, \dots, U_j \\ \psi_{ju,w+1} = 0, & \forall j, u = 1, 2, \dots, U_j, w = 1, 2, \dots, W_{ju} - 1 \end{cases} \quad (11)$$

Then, the subsequent state can be written as $s' = G_a^s + \Psi$. To keep the notations simple, we introduce P_Ψ in the following equation:

$$p(s, a, s') = \begin{cases} \prod_{u=1}^{U_j} p(\tilde{n}_{ju} = \psi_{ju,1}) = P_\Psi, & \text{if } s' = G_a^s + \Psi \\ 0, & \text{if } s' \neq G_a^s + \Psi \end{cases} \quad (12)$$

Then, we have

$$\sum_{s' \in S} p(s, a, s') V^{\pi^*}(s') = \sum_{\Psi=0}^{+\infty} P_\Psi V^{\pi^*}(G_a^s + \Psi) \quad (13)$$

With these notations and analysis, we propose the following properties of $V^{\pi^*}(s)$:

Proposition 2. (i) $V^{\pi^*}(s)$ is increasing in s .
(ii) $V^{\pi^*}(s)$ is increasing in $P(s)$.

Proof. See Appendix C. □

Proposition 2 indicates that the properties of the single-period optimal cost $C_0(s)$ proposed in (i) and (iii) of Proposition 1 also hold for the optimal value function $V^{\pi^*}(s)$. Considering that $V^{\pi^*}(s)$ is monotonically increasing in s and $P(s)$, we can prove the following properties of $\pi^*(s)$ and $Q^{\pi^*}(s, a)$ that enable us to simplify the exploration of the action space.

Proposition 3. For any patient type $\{juw\}$, if $(c_d - c_b)v_juw > c_o\bar{d}_j + c_e\bar{l}_j$ holds, then all the type- $\{juw\}$ patients are scheduled by $\pi^*(s)$.

Proof. See Appendix D. □

Proposition 4. $\forall s \in S$: $Q^{\pi^*}(s, a)$ is decreasing in $P(a)$.

Proof. See Appendix E. □


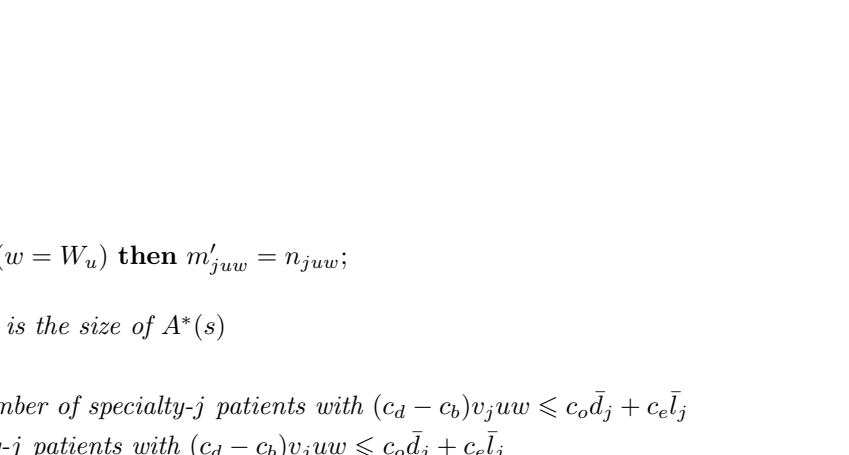
Based on Proposition 3, the procedure of searching for the greedy action $a(s)$ for state s can be simplified by excluding the actions in which the patients satisfying $(c_d - c_b)v_juw > c_o\bar{d}_j + c_e\bar{l}_j$ or $w = W_{ju}$ are not all scheduled. Moreover, as Proposition 4 suggests that the Q-value is decreasing in $P(a)$, the action set can be further narrowed down: for any $a \in A(s)$, if $\exists a' \in A(s)$ s.t. $P(a') > P(a)$, then $Q^{\pi^*}(s, a) > Q^{\pi^*}(s, a')$ holds by Proposition 4, hence a is not the greedy action and can be neglected. Let $A^*(s) \subseteq A(s)$ be the set of actions that needs to be evaluated, then the procedure of determining $A^*(s)$ is provided in Algorithm 1. In this paper, we do not consider the rare case with $U_j = W_{ju} = 1$ and $((c_d - c_b)v_juw \leq c_o\bar{d}_j + c_e\bar{l}_j$ for all j, u , and w ; thus, when solving the MDP model proposed in Section 3, we can integrate Algorithm 1 into most solution approaches, including DP-based algorithms and simulation-based ADP algorithms, in which the greedy action of each visited state should be found and the efficiency of exploring the action space can be significantly improved.

5 Solution approach

The algorithm proposed in the previous section (Algorithm 1) can reduce the number of actions to be evaluated for each state, thus facilitating the exploration of the action space. However, it cannot be independently employed to solve the MDP model proposed in Section 3. As an acceleration technique, Algorithm 1 should always be integrated into a solution approach of MDP, such as PI, VI, and RTDP. For the patient admission control problem studied in this paper, there are $\sum_{j=1}^J \sum_{u=1}^{U_j} W_{ju}$ different patient types, and the dimension of the MDP model's state space is

Algorithm 1: Procedure of determining $A^*(s)$

Input: state $s = \{n_{juw}\}$
Output: action set $A^*(s)$

- 1 Initialize $a' = \{m'_{juw}\} = \mathbf{0}$;
- 2 **for** $j = 1, 2, \dots, J$ **do**
- 3 **for** $u = 1, 2, \dots, U_j$ **do** 
- 4 **for** $w = 1, 2, \dots, W_{ju}$ **do**
- 5 **if** $((c_d - c_b)v_{juw} > c_o\bar{d}_j + c_e\bar{l}_j) \vee (w = \underline{W_u})$ **then** $m'_{juw} = n_{juw}$; 
- 6 Let $A^*(s) = \{a'\}$ and $I = \|A^*(s)\| = 1$; *//I is the size of $A^*(s)$*
- 7 **for** $j = 1, 2, \dots, J$ **do**
- 8 $N_j = \sum_{u=1}^{U_j} \sum_{w=1}^{W_{ju}} (n_{juw} - m'_{juw})$; *//number of specialty-j patients with $(c_d - c_b)v_{juw} \leq c_o\bar{d}_j + c_e\bar{l}_j$*
- 9 $M_j = 0$; *//number of scheduled specialty-j patients with $(c_d - c_b)v_{juw} \leq c_o\bar{d}_j + c_e\bar{l}_j$*
- 10 $I = I \times (N_j + 1)$; *//compute the size of $A^*(s)$*
- 11 **for** $i = 1, 2, \dots, I$ **do**
- 12 $a_i = \{m_{juw}\} = a'$;
- 13 *//exhaust all the possible combinations of M_j*
- 14 **for** $j = 1, 2, \dots, J$ **do**
- 15 **if** $M_j < N_j$ **then**
- 16 $M_j = M_j + 1$;
- 17 **foreach** integer $k \in [1, j - 1]$ **do** $M_k = 0$;
- 18 **break**;
- 19 *//for each combination of M_j , find the action with the maximum value of $P(a)$*
- 20 **for** $j = 1, 2, \dots, J$ **do**
- 21 $\{u', w'\} = \arg \max_{((c_d - c_b)v_{juw} \leq c_o\bar{d}_j + c_e\bar{l}_j) \wedge (w \neq \underline{W_u})} v_{juw}$;
- 22 $x = M_j$;
- 23 **while** *true* **do**
- 24 **if** $n_{ju'w'} \geq x$ **then**
- 25 $m_{ju'w'} = x$;
- 26 **break**;
- 27 **else**
- 28 $m_{ju'w'} = n_{ju'w'}$;
- 29 $x = x - n_{ju'w'}$;
- 30 $\{u', w'\} = \arg \max_{(uw \leq u'w') \wedge (w \neq \underline{W_u}) \wedge ((u \neq u') \vee (w \neq w'))} v_{juw}$;
- 31 $A^*(s) = A^*(s) \cup \{a_i\}$;

increasing in $\sum_{j=1}^J \sum_{u=1}^{U_j} W_{ju}$ at an exponential rate. Moreover, as there can be large numbers of $\sum_{j=1}^J U_j$ different types of newly arrived patients in each week, the number of possible subsequent states for any state-action pair is enormous, i.e., the outcome space is huge as well (exponentially increasing in $\sum_{j=1}^J U_j$). Therefore, even with the acceleration of Algorithm 1, solving the MDP model using traditional DP-based algorithms is still computationally intractable. To cope with the curses of dimensionality in the state space and the outcome space, in this section, we

propose an ADP algorithm based on reinforcement learning.

The basic idea of the ADP algorithm is to reduce the model dimensionality by approximating the value function. It is well known that value function approximation can either be parametric or non-parametric: the former adopts a low-dimensional linear function to approximate the true optimal value function $V^{\pi^*}(s)$, while the latter does not rely on functional structures but directly approximates the values of observed instances (Ulmer & Thomas, 2020). A comprehensive overview of the two value function approximation methods is provided by Powell (2011). Non-parametric methods often need to aggregate the state space and use lookup tables which may suffer from the curse of dimensionality (Powell & Meisel, 2015; Voelkel et al., 2020). In comparison, parametric methods are easier to compute and more suitable for online computation (Powell & Meisel, 2015). In the ADP algorithm proposed in this paper, we adopt a parametric method that approximates the true optimal value function $V^{\pi^*}(s)$ using a classical feature-based linear function:

$$\hat{V}(s, \Theta) = \underbrace{\Phi^T(s)}_{\text{feature vector}} \Theta = \sum_{\xi=1}^{\Xi} \underbrace{\phi_{\xi}(s)}_{\text{feature}} \theta_{\xi} \approx V^{\pi^*}(s) \quad (14)$$

In (14), $\phi_{\xi}(s)$ captures the ξ th feature of state s and each state $s \in S$ has a corresponding feature vector $\Phi(s) = [\phi_1(s), \dots, \phi_{\xi}(s), \dots, \phi_{\Xi}(s)]^T$. For our MDP model, it is straightforward to define that each component n_{juw} of state s corresponds to a feature $\phi_{\xi}(s)$, thus the length of feature vector $\Phi(s)$ is $\Xi = \sum_{j=1}^J \sum_{u=1}^{U_j} W_{ju}$. $\Theta = [\theta_1, \dots, \theta_{\xi}, \dots, \theta_{\Xi}]^T$ in (14) is a parameter vector of length Ξ and each component θ_{ξ} is a tunable parameter. We want to compute the optimal parameter vector Θ^* which minimizes the gaps between $\hat{V}(s, \Theta)$ and $V^{\pi^*}(s)$:

$$\longrightarrow \quad \Theta^* = \arg \min_{\Theta \in \mathbb{R}^{\Xi}} \sum_{s \in S} \left[\hat{V}(s, \Theta) - V^{\pi^*}(s) \right]^2 \quad (15)$$

With Θ^* , we can compute a near-optimal policy based on the approximate value function $\hat{V}(s, \Theta^*)$. As the computation of the complex value function $V^{\pi^*}(s)$ in the high-dimensional state space S is replaced by the computation of the low-dimensional parameter vector Θ^* , the curse of dimensionality in the state space can be drastically alleviated.

Many reinforcement learning methods can be used to compute Θ^* (by iteratively updating Θ), such as temporal difference learning (TD(λ): Sutton, 1988; Tsitsiklis & Van Roy, 1997), least-squares temporal difference learning (LS-TD(λ): Barto et al., 1995; Boyan, 2002), and recursive least-squares temporal difference learning (RLS-TD(λ): Barto et al., 1995; Xu et al., 2002). In order to combine reinforcement learning with ADP, we adopt an on-policy learning strategy introduced by Powell (2011), under which the parameter vector Θ and the policy based on it are updated simultaneously and all the computation is performed online.

Before making any decision, we first initialize Θ arbitrarily, e.g., $\Theta_0 = \mathbf{0}$. Then, at each decision epoch (week) τ , we randomly sample the stochastic information (i.e., patient arrivals) of the next N epochs. At each simulated epoch n , we select action a_n for the current state s_n (note that $s_1 = s_{\tau}$) according to the approximate value function with the latest parameter vector Θ_{n-1} :

$$a_n = \arg \min_{a \in A^*(s_n)} \left[\underbrace{C(s_n, a)}_{\text{cost}} + \gamma \Phi^T(G_a^{s_n} + \Psi_n^a) \Theta_{n-1} \right] \quad (16)$$

where $G_a^{s_n}$ is the post-action state of state-action pair (s_n, a) ; Ψ_n^a is a vector of the numbers of new patients (defined in Section 4) and it is independently and identically sampled from Poisson distributions for each action a and each epoch n . Then, we update Θ_{n-1} to Θ_n and compute the next state to be visited $s_{n+1} = G_{a_n}^{s_n} + \Psi_n^{a_n}$. In this way, we can obtain a trajectory $(s_1, a_1, \Psi_1^{a_1}, s_2, a_2, \Psi_2^{a_2}, \dots, s_N, a_N, \Psi_N^{a_N})$ with $s_1 = s_{\tau}$.

The process described above can be repeated multiple times for the same state s_{τ} , until some termination criterion is satisfied. That is, at each real decision epoch τ , we generate $M \geq 1$ trajectories with independently and

identically sampled vectors Ψ . Thus, parameter vector Θ and the policy based on it are updated $M \times N$ times before we determine the action to be executed in week τ :

$$a_\tau = \arg \min_{a \in A^*(s_\tau)} [C(s_\tau, a) + \gamma \Phi^T(G_a^{s_\tau} + \bar{\Psi})\Theta_\tau] \quad (17)$$

where vector $\bar{\Psi}$ consists of the expected arrival numbers \bar{n}_{ju} of $\sum_{j=1}^J U_j$ patient types, and Θ_τ is the latest updated parameter vector. From (16) and (17), it can be seen that all the actions we take in simulations (a_n) and in the real world (a_τ) are selected according to the approximate value function $\hat{V}(s, \Theta)$. This is the so-called on-policy learning strategy. An advantage of this strategy is that we explore the outcome space along randomly sampled trajectories, rather than evaluating all the possible subsequent states of each state-action pair. In this way, the curse of dimensionality in the outcome space can be effectively solved.

Next, we briefly present the reinforcement learning method integrated into the ADP algorithm for updating Θ . After action a_n is computed by (16) at simulated epoch n , we compute the gap between two successive value function approximations, i.e., the temporal difference or TD error:

$$e_n = C(s_n, a_n) + \gamma \hat{V}(s_{n+1}, \Theta_{n-1}) - \hat{V}(s_n, \Theta_{n-1}) = C(s_n, a_n) - [\Phi(s_n) - \gamma \Phi(s_{n+1})]^T \Theta_{n-1} \quad (18)$$

The TD(λ) method introduced by Sutton (1988) updates parameter vector Θ using gradient descent:

$$\rightarrow \Theta_n = \Theta_{n-1} + \eta_n e_n \sum_{k=1}^n (\gamma \lambda)^{n-k} \nabla \hat{V}(s_k, \Theta_{n-1}) \quad (19)$$

where η_n is a sequence of step-size parameters, $\lambda \in [0, 1]$ is a decaying factor, and $\nabla \hat{V}(s_k, \Theta_{n-1}) = \Phi(s_k)$ is the vector of partial derivatives with respect to the components of Θ_{n-1} . Equation (19) can be simplified by defining z_n as the eligibility trace which implicitly stores the history of a trajectory (Tsitsiklis & Van Roy, 1997):

$$\rightarrow z_n = \sum_{k=1}^n (\gamma \lambda)^{n-k} \nabla \hat{V}(s_k, \Theta_{n-1}) = \sum_{k=1}^n (\gamma \lambda)^{n-k} \Phi(s_k) \quad (20)$$

Thus, TD(λ) can be represented by the following recursive equations:

$$\begin{cases} z_n = \Phi(s_n) + \gamma \lambda z_{n-1} \\ \Theta_n = \Theta_{n-1} + \eta_n e_n z_n \end{cases} \quad (21)$$

where z is often arbitrarily initialized, e.g., $z_0 = \mathbf{0}$. The convergence with probability one of TD(λ) has been established by Tsitsiklis and Van Roy (1997). However, TD(λ) is particularly impacted by the step-size parameters η_n , as a poor choice of η_n can drastically slow down the convergence (Barto et al., 1995). To eliminate η_n , Boyan (2002) extends TD(λ) to LS-TD(λ) based on the fact that the changes of Θ made by TD(λ) are in the following form:

$$\Theta_n = \left\{ \sum_{k=1}^n z_k [\Phi(s_k) - \gamma \Phi(s_{k+1})]^T \right\}^{-1} \left[\sum_{k=1}^n C(s_k, a_k) z_k \right] = \mathcal{A}_n^{-1} b_n \quad (22)$$

It can be seen from (22) that step-size parameters η_n are eliminated, but the inverse of a $\Xi \times \Xi$ matrix \mathcal{A}_n has to be computed at each time epoch, hence the computational complexity of LS-TD(λ) is $O(\Xi^3)$.

Further, Xu et al. (2002) improve LS-TD(λ) using the recursive least-squares (RLS) method of Ljung and Söderström (1983). Since the RLS-TD(λ) method proposed by Xu et al. (2002) has a lower computational complexity $O(\Xi^2)$, we employ RLS-TD(λ) to update Θ in this paper. To facilitate the computation, RLS-TD(λ) defines a $\Xi \times \Xi$

variance matrix $\mathcal{P}_n = \mathcal{A}_n^{-1} = \{\mathcal{A}_{n-1} + z_n[\Phi(s_n) - \gamma\Phi(s_{n+1})]^T\}^{-1}$. By Lemma 2.1 of Ljung and Söderström (1983), \mathcal{P}_n can be written in the following recursive form:

$$\mathcal{P}_n = \mathcal{P}_{n-1} - \mathcal{P}_{n-1}z_n \{1 + [\Phi(s_n) - \gamma\Phi(s_{n+1})]^T \mathcal{P}_{n-1} z_n\}^{-1} [\Phi(s_n) - \gamma\Phi(s_{n+1})]^T \mathcal{P}_{n-1} \quad (23)$$

Then, using the RLS method with (22) and (23), Θ can be updated in the following way:

$$\Theta_n = \Theta_{n-1} + \frac{\mathcal{P}_{n-1}z_n e_n}{1 + [\Phi(s_n) - \gamma\Phi(s_{n+1})]^T \mathcal{P}_{n-1} z_n} \quad (24)$$

Combining (18), the first line of (21), (23), and (24), we have the following update rules of RLS-TD(λ):

$$\begin{cases} e_n = C(s_n, a_n) - [\Phi(s_n) - \gamma\Phi(s_{n+1})]^T \Theta_{n-1} \\ z_n = \gamma\lambda z_{n-1} + \Phi(s_n) \\ \mathcal{P}_n = \mathcal{P}_{n-1} - \frac{\mathcal{P}_{n-1}z_n[\Phi(s_n) - \gamma\Phi(s_{n+1})]^T \mathcal{P}_{n-1}}{1 + [\Phi(s_n) - \gamma\Phi(s_{n+1})]^T \mathcal{P}_{n-1} z_n} \\ \Theta_n = \Theta_{n-1} + \frac{\mathcal{P}_{n-1}z_n e_n}{1 + [\Phi(s_n) - \gamma\Phi(s_{n+1})]^T \mathcal{P}_{n-1} z_n} \end{cases} \quad (25)$$

where \mathcal{P}_n is initialized as $\mathcal{P}_0 = \beta\mathcal{I}$ (β is a positive real number and \mathcal{I} is the identity matrix).

Based on the methods discussed above, including parametric value function approximation (14), on-policy learning, and RLS-TD(λ), we develop the ADP algorithm presented in Algorithm 2. The depth of each sampled trajectory is fixed to a positive integer N . In each week, the process of sampling new trajectories terminates when the relative improvement of Θ is lower than a threshold $\epsilon > 0$ (line 12). Before making the first decision in the first week $\tau = 1$, all the components of z_0 , \mathcal{P}_0 , and Θ_0 are initialized as 0. Then, they are continuously updated in all the following weeks. In line 5 and 14 of Algorithm 2, we assume that Algorithm 1 is used to reduce the actions to be evaluated. If Algorithm 1 is not used, line 5 and 14 should be skipped and the reduced action spaces $A^*(s_n)$ and $A^*(s_\tau)$ in Algorithm 2 should be replaced with the full action spaces $A(s_n)$ and $A(s_\tau)$, respectively.

6 Experimental results

We conduct numerical experiments on different test problems to validate the efficiency and accuracy of the proposed algorithms. All the programs are coded in C++ and run on a PC with an Intel(R) Core(TM) i7-3770 CPU @3.40GHz processor and a RAM of 8GB. In Section 6.1, the algorithms proposed in this paper and several exact methodologies are employed to solve a small-sized test problem. We compare the computational performances of these solution approaches before examining the sensitivity of the resulting policy and the CPU time to the problem parameters. In Section 6.2, we solve a large-sized single-specialty test problem and perform sensitivity analyses for the key parameters of the RLS-TD(λ)-based ADP algorithm. Finally, we employ the proposed algorithms to solve a realistically sized multi-specialty patient admission control problem and present the results in Section 6.3.

The quality of each policy computed in this section is evaluated through a numerical simulation with a finite time length τ_{max} . In each simulated week, the numbers of different types of newly arrived patients are randomly sampled from Poisson distributions with constant arrival rates \bar{n}_{ju} . The values of τ_{max} and \bar{n}_{ju} are different for different test problems. When multiple policies are computed for the same test problem, the value of τ_{max} and the sequences of patient arrivals $(\Psi_1, \Psi_2, \dots, \Psi_{\tau_{max}})$ are identical for the simulations of these policies. Each patient's surgery duration and LOS are realized when he/she is scheduled for surgery. The cost of each week is computed by taking the average value of cost function (4) in a large number of scenarios. Each scenario contains the realizations of all the scheduled

Algorithm 2: ADP algorithm based on RLS-TD(λ) learning

Input: s_τ , $z_{\tau-1}$, $\mathcal{P}_{\tau-1}$, and $\Theta_{\tau-1}$

- 1 Initialize $s_1 = s_\tau$, $z_0 = z_{\tau-1}$, $\mathcal{P}_0 = \mathcal{P}_{\tau-1}$, and $\Theta_0 = \Theta_{\tau-1}$;
- 2 **while** *true* **do**
- 3 // *A new trial (trajectory) starts here;*
- 4 **for** $n = 1, 2, \dots, N$ **do**
- 5 Determine $A^*(s_n)$ by Algorithm 1; // *This step can be skipped;*
- 6 **foreach** $a \in A^*(s_n)$ **do**
- 7 Randomly sample Ψ_n^a from Poisson distributions;
- 8 Compute a_n by (16);
- 9 Compute $s_{n+1} = G_{a_n}^{s_n} + \Psi_n^{a_n}$;
- 10 Update z_n , \mathcal{P}_n , and Θ_n by (25);
- 11 // *A trial (trajectory) ends here;*
- 12 **if** $\|\frac{\Theta_n - \Theta_0}{\Theta_0}\| < \epsilon$ **then** break;
- 13 Let $s_1 = s_\tau$, $z_0 = z_n$, $\mathcal{P}_0 = \mathcal{P}_n$, and $\Theta_0 = \Theta_n$;
- 14 Determine $A^*(s_\tau)$ by Algorithm 1; // *This step can be skipped.*
- 15 Let $z_\tau = z_n$, $\mathcal{P}_\tau = \mathcal{P}_n$, and $\Theta_\tau = \Theta_n$;
- 16 Compute a_τ by (17);

Output: a_τ , z_τ , \mathcal{P}_τ , and Θ_τ

patients' surgery durations and LOSs, which are randomly sampled from lognormal distributions. We arbitrarily set the scenario number to 10000.

The notations used in this section for measuring the computational performances of the employed algorithms are listed in Table 1.

Table 1: Definitions of variables used in Section 6

Variable	Definition
T	total CPU time (s)
t	CPU time consumed in one week (ms)
ω_{ju}	waiting time of specialty- j patients with urgency coefficient u (week)
o	over-utilization of ORs in all specialties during one week (h)
o_j	over-utilization of ORs in specialty j during one week (h)
e	shortage of SICU recovery beds during one week (bed-day)
c	total cost of one week
c_p	patient-related cost of one week
c_h	hospital-related cost of one week
\bar{x}	mean of variable x
$\sigma(x)$	standard deviation of variable x
$\ A\ $	total number of feasible actions for all the visited states
$\ A^*\ $	total number of actually evaluated actions

6.1 Algorithm comparisons and sensitivity analyses of problem parameters

In this subsection, we conduct numerical experiments for an arbitrarily generated test problem. The proposed ADP algorithm is compared to two DP-based algorithms: VI and VPI-RTDP, whose complete procedures can be found in Mausam and Kolobov (2012) and Sanner et al. (2009), respectively. For each algorithm employed, we search for the greedy actions in two different ways: by evaluating the action sets $A^*(s)$ determined by Algorithm 1 and by enumerating the entire action sets $A(s)$. We use a superscript $*$ to distinguish the algorithms that search for the greedy action in the former way from the others. For example, VI * searches for the greedy actions in the action sets $A^*(s)$ determined by Algorithm 1, while VI evaluates all the feasible actions in $A(s)$. Due to the low computational efficiency of the DP-based algorithms, the test problem solved in this subsection is much smaller than the realistically sized ones.

The small-sized test problem is a two-specialty patient admission control problem. The relative importance of the two specialties $j = 1, 2$ are $[v_1, v_2] = [1, 2]$. Patients of the two specialties are divided into two urgency groups $u = 1, 2$, and their maximum allowed waiting times are $[W_{11}, W_{12}, W_{21}, W_{22}] = [4, 2, 3, 2]$ weeks. New patient arrivals are assumed to follow Poisson distributions, with the parameters $[\tilde{n}_{11}, \tilde{n}_{12}, \tilde{n}_{21}, \tilde{n}_{22}] = [1.0, 0.5, 0.25, 0.25]$. Considering that the DP-based algorithms are unable to tackle infinite state spaces, we truncate the Poisson distributions by omitting the values whose probabilities are lower than 0.005; thus $\tilde{n}_{11} < 5$, $\tilde{n}_{12} < 4$, $\tilde{n}_{21} < 3$, and $\tilde{n}_{22} < 3$. The size of the state space of this problem can thereby be calculated as $|S| = 5^4 \times 4^2 \times 3^3 \times 3^2 = 2.43 \times 10^6$. Surgery durations and LOSs follow lognormal distributions, with $[\bar{d}_1, \bar{d}_2] = [2, 4]$ hours and $[\bar{l}_1, \bar{l}_2] = [4, 2]$ days, and the standard deviations of surgery durations and the LOS for each specialty are assumed to be equal to their means. The unit costs of the small-sized test problem are determined as $[c_b, c_d] = [50, 100]$, $c_o = 400$ per hour and $c_e = 1000$ per patient day. The regular OR capacities for the two specialties are $[B_1, B_2] = [3, 2]$ hours, and the available SICU capacity is $R = 7$ bed-days per week. The estimated availability rates of ORs and the SICU are arbitrarily determined as $\rho_1 = \rho_2 = 1$ in the small-sized test problem.

Using the Monte-Carlo simulation method, we randomly sample arrival information over 1,000 weeks, as well as sampling the surgery durations and LOSs of all the arrived patients. Using these samples, we carry out a numerical simulation for each solution approach for 1,000 consecutive weeks. The parameters of the ADP algorithm are arbitrarily set as $\lambda = 0$, $\beta = 1$, $N = 5000$ and $\epsilon = 0.0001$ in this subsection, while the discount factor of the MDP model is $\gamma = 0.99$. The computational results are presented in Table F.1 of Appendix F.

From Table F.1, it can be observed that the RLS-TD(λ)-based ADP algorithm is the most efficient solution approach. The CPU time consumed by ADP is only 0.02% and 0.26% of that consumed by VI and VPI-RTDP, respectively. Moreover, applying Algorithm 1 drastically improves the computational efficiency of all the employed solution approaches, and does not lead to any policy deterioration. To be specific, when applied to VI, VPI-RTDP, and ADP, Algorithm 1 reduces CPU time by 92.00%, 73.33%, and 14.28%, respectively. Table F.1 also indicates that the resulting policies of the algorithms employed are similar. The relative gap between the total costs of ADP/ADP * and those of the DP-based algorithms is below 3.16%. Among the DP-based algorithms, we can observe that the total costs of VI/VI * are slightly higher than those of VPI-RTDP/VPI-RTDP * , implying that the policies of VI/VI * are slightly further from convergence than those of VPI-RTDP/VPI-RTDP * . Obviously, the former can be further improved by modifying the user-defined stopping criterion, but doing so requires even more CPU time. In summary, the RLS-TD(λ)-based ADP algorithm provides a high-quality, near-optimal policy for the problem under study, and Algorithm 1 is capable of accelerating the employed algorithms without any deterioration of the policy.

Next, we perform sensitivity analyses to evaluate the effects of the problem parameters on the resulting policy and the computational complexity. We use an exact algorithm (VPI-RTDP *) and the proposed ADP * algorithm to solve the small-sized two-specialty test problem with different values for the problem parameters c_d , c_o , c_e , B_1 ,

B_2 , and R . The results of the sensitivity analyses are demonstrated in Figure 2 to 7. It can be seen from these figures that the ADP* policy is very close to that of VPI-RTDP* in most cases, despite variations in the problem parameters; again, this validates the accuracy of the ADP* algorithm. Specifically, Figure 2 shows that changing the unit waiting cost c_d (or the ratio of c_d to c_b) only leads to slight changes in the resulting policy. In contrast, Figure 3 and 4 illustrate that the hospital-related cost c_h and the total cost c are increasing in both the unit OR overtime cost c_o and the unit SICU excess cost c_e , while Figure 5, 6, and 7 show that c_h and c are decreasing in the OR capacity B_1 , B_2 , and the SICU capacity R . From Figure 2 to 7, we can also observe that the patient-related cost c_p remains relatively stable as the problem parameters change, which implies that the resulting policy is not very sensitive to the problem parameters. In terms of computational complexity, Figure 2 to 7 show that the CPU time consumed by VPI-RTDP* is increasing in c_o , c_e and decreasing in c_d , B_2 , R , while the CPU time for ADP* does not change too much as the problem parameters vary. Additionally, the CPU time for ADP* is much less than that of VPI-RTDP* in all cases.

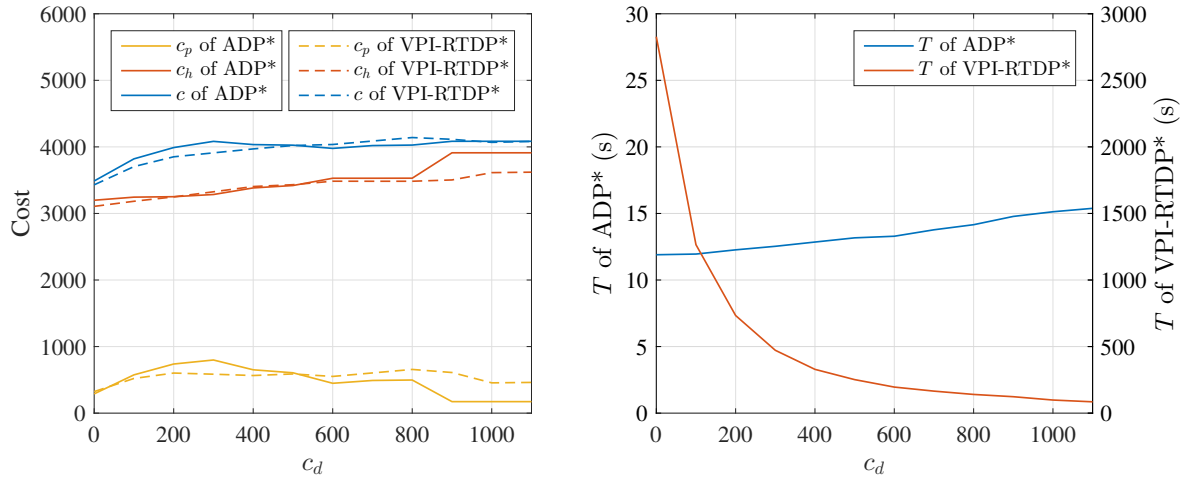


Figure 2: Sensitivity analysis for the unit waiting cost c_d

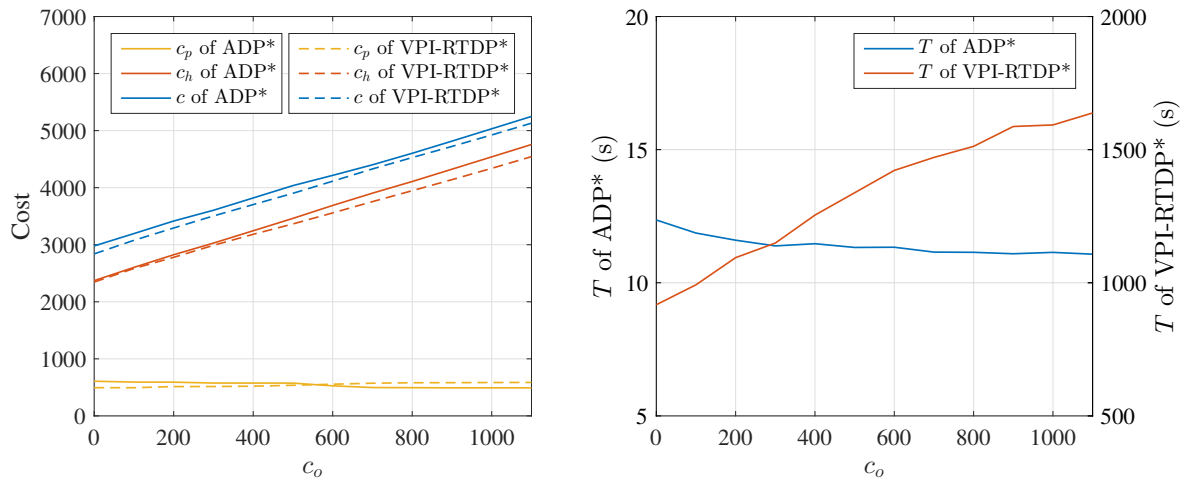


Figure 3: Sensitivity analysis for the unit cost c_o of overusing ORs

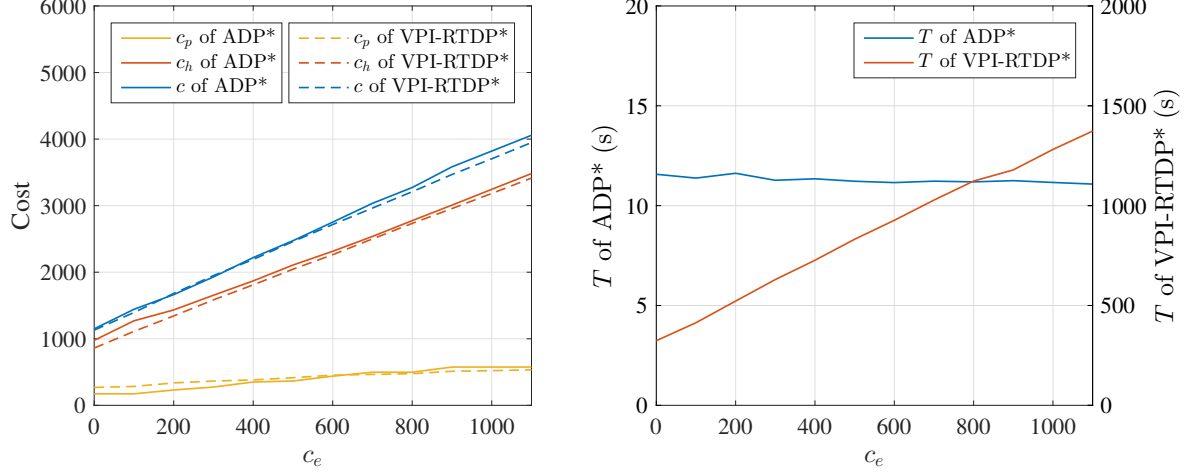


Figure 4: Sensitivity analysis for the unit cost c_e of exceeding SICU capacity

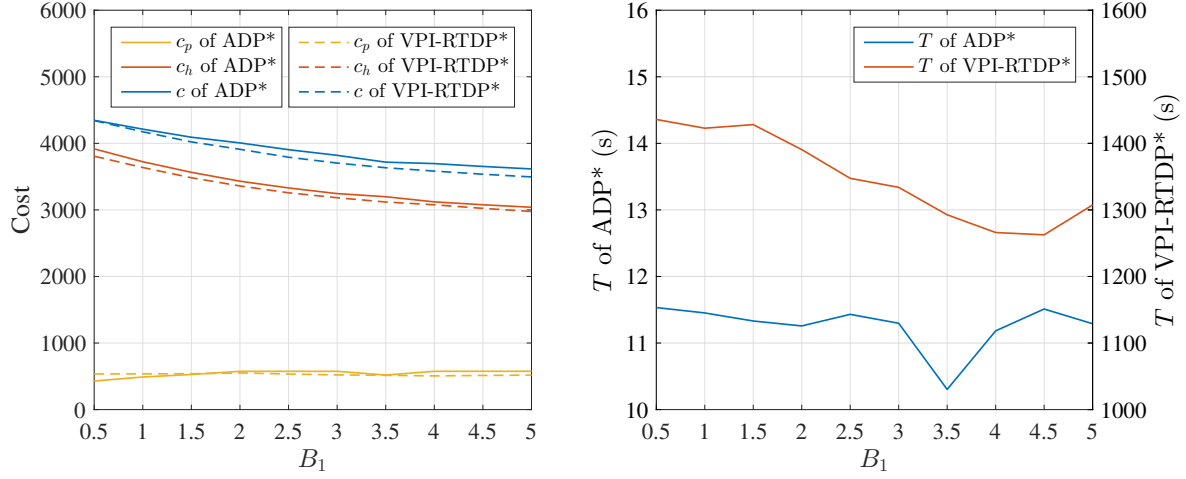


Figure 5: Sensitivity analysis for the OR capacity B_1 of specialty 1

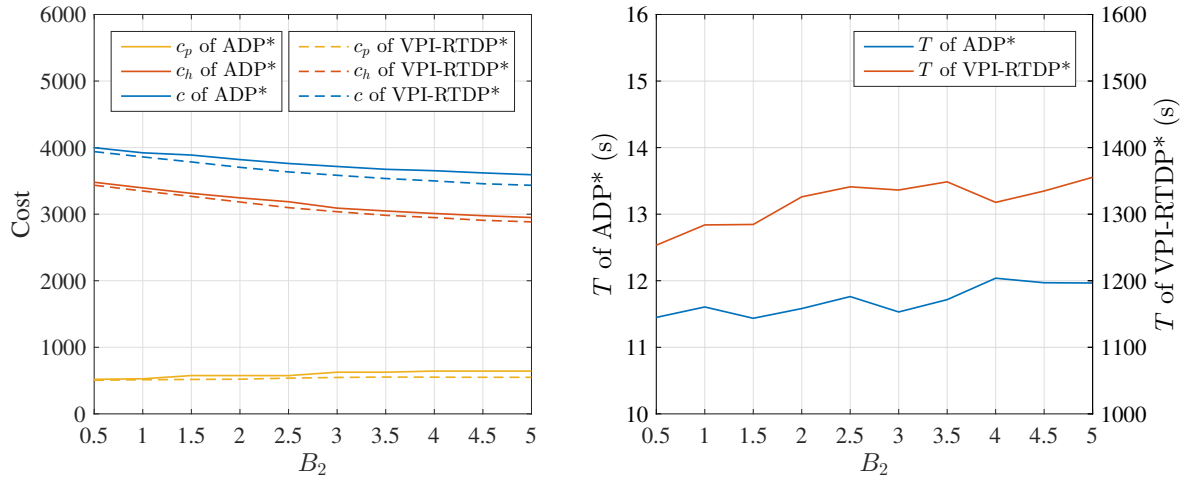


Figure 6: Sensitivity analysis for the OR capacity B_2 of specialty 2

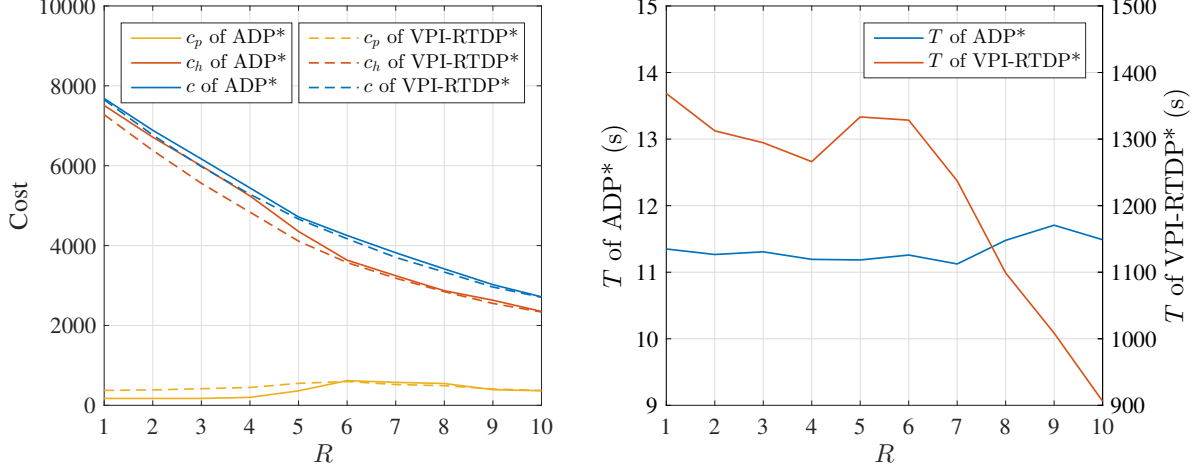


Figure 7: Sensitivity analysis for the SICU capacity R

6.2 Sensitivity analyses of the ADP algorithm parameters

The experimental results of the previous subsection reveal that the proposed RLS-TD(λ)-based ADP algorithm is distinctly more efficient than the conventional DP-based algorithms. At the same time, the gaps between the resulting costs of these algorithms are insignificant. To further evaluate the capability of the proposed algorithms to cope with large cases, we employ the ADP* algorithm (i.e., the RLS-TD(λ)-based ADP algorithm in combination with the greedy action searching method presented by Algorithm 1) to solve a realistically sized single-specialty patient admission control problem, and perform sensitivity analyses for the key parameters λ and β of the ADP/ADP* algorithm.

The problem to be solved in this subsection is the admission control of coronary artery bypass grafting (CABG) surgeries, which has previously been studied by Min and Yih (2010a). Our work differs from Min and Yih (2010a), however, as it incorporates patient waiting times, dynamic patient priority scores, and due dates into the MDP model. The problem settings are based on the configurations of Min and Yih (2010a) and the real-life data provided by Sobolev and Kuramoto (2008). Patients from the same specialty ($j = 1$) are divided into three urgency groups, with $u = 1, 2, 6$, $[W_{11}, W_{12}, W_{16}] = [12, 6, 2]$, $[\bar{n}_{11}, \bar{n}_{12}, \bar{n}_{16}] = [3, 5, 1]$, $\tilde{n}_{11} \leq 9$, $\tilde{n}_{12} \leq 13$ and $\tilde{n}_{16} \leq 5$. The size of the state space is as large as $10^{12} \times 14^6 \times 6^2 \approx 2.71 \times 10^{20}$. Parameters of lognormal distributions for surgery durations and LOSs are $[\bar{d}_1, \sigma(d_1)] = [4, 1.72]$ hours and $[\bar{l}_1, \sigma(l_1)] = [2, 2]$ days, respectively. Costs are determined as $[c_b, c_d, c_o, c_e] = [100, 150, 1500, 1500]$. Available surgical resources are $B_1 = 40$ hours and $R = 25$ bed-days, with estimated availability rates $[\rho_1, \rho_2] = [0.9, 0.72]$.

Simulations for 1,000 consecutive weeks are performed to address the CABG surgery admission control problem. As a result of the increasing problem scale, DP-based algorithms are no longer capable of computing the optimal policy within an acceptable CPU time. Therefore, current practice is to compute a myopic policy in which each action only minimizes the cost of the present week. This policy serves as the benchmark for the resulting ADP* policy and can easily be obtained by solving the MDP model with discount factor $\gamma = 0$. Since the action spaces of realistic problems are considerably large, we employ Algorithm 1 to reduce the number of actions to be evaluated. The simulation results for the myopic policy, as well as the resulting ADP* policies with $N = 1000$ and $\epsilon = 0.001$, are listed in Table F.2 in Appendix F.

From the data presented in Table F.2, we can see that all the simulations can be finished within a reasonable CPU time, and that computing the myopic policy ($\gamma = 0.00$) is much easier than computing the ADP* policy ($\gamma = 0.99$);

that is, the CPU time consumed by the former ($\bar{t} = 0.011\text{ms}$) is much less than that of the latter ($\bar{t} \in (62, 135)\text{ms}$). However, the ADP* policy significantly outperforms the myopic policy in terms of reducing the total costs and the patients' waiting times. In comparison to the ADP* policy, which minimizes the expected total costs over an infinite horizon, the myopic policy optimizes the cost of each week separately, without considering the inter-week correlations. As a result, it generally schedules fewer patients than does the ADP* policy and thereby leads to longer waiting times. To be specific, the average waiting times of patients with urgency coefficients 1, 2 and 6 under the ADP* policy are lower than those under the myopic policy by $31.4 \pm 21.4\%$, $33.1 \pm 21.7\%$ and $11.9 \pm 4.0\%$, respectively. As the ADP* policy tends to schedule more patients than does the myopic policy, it increases the over-utilizations of the ORs and SICU by 0.79 ± 0.51 hours per week and 0.29 ± 0.27 patient-days per week, respectively. However, the total cost of the ADP* policy is $26.8 \pm 14.6\%$ lower than that of the myopic policy. Table F.2 also illustrates the effects of the values of λ and β on the resulting policy. It can be observed that the patients' waiting times and the total cost are decreasing in λ and β ; however, the over-utilizations of surgical resources are increasing in these parameters.

We record the sizes of $A(s)$ and $A^*(s)$ for each visited state in the simulations for the CABG surgery admission control problem. We find that the sizes of $A(s)$ of some states can be as large as 1.59×10^{11} , while the sizes of $A^*(s)$ determined by Algorithm 1 are no larger than 75. Table F.2 also shows that the ratio of $\|A^*\|$ to $\|A\|$ in each simulation is lower than 5%, which is significantly lower than the values of $\|A^*\|/\|A\|$ for the small-sized test problem solved in Section 6.1. This fact implies that Algorithm 1 leads to greater improvements in computational efficiency for larger problems.

To further explore how the scheduling policy can be influenced by the key parameters of the ADP* algorithm, we analyze the sensitivity of several performance measures to the variances of λ and β . Figure 8 demonstrates the experimental results for the CABG problem; it indicates that λ varies from 0.0 to 1.0 and β varies from 10^{-5} to 10^5 . It can be observed that, when λ exceeds 0.8, the patients' waiting times and the total cost sharply decrease, regardless of the value of β ; meanwhile, the over-utilizations of the ORs and SICU increase significantly. When $\lambda \leq 0.8$, the larger value of β results in lower cost and shorter waiting times for patients. However, it also leads to greater overuse of the ORs and SICU. More importantly, no matter how the values of λ and β vary, the average cost per week \bar{c} of the ADP* policy is always significantly lower than that of the myopic policy (refer to Table F.2).

Figure 9 compares the evolutions of the parameter vector Θ during the simulations, using different values for λ and β . The length of vector Θ for the CABG problem is $\Xi = 20$; we arbitrarily select the values of θ_1 and θ_{11} to be demonstrated in Figure 9. The first row of Figure 9 shows that the smaller value of β leads to a lower convergence rate, and that the values of θ_1 and θ_{11} converge especially slowly when $\beta = 10^{-5}$. The second row of Figure 9 reveals that the values to which parameters θ_ξ converge are increasing in line with the value of λ , which explains why different values of λ result in different policies (refer to Table F.2 and Figure 8). In terms of computational efficiency, Figure 10(a) shows that the CPU time goes up as λ increases from 0.0 to 0.8, which implies that larger values of λ require more CPU time before the convergence of Θ occurs. When $0.8 \leq \lambda \leq 1.0$, the resulting policy tends to schedule more patients in each decision; we can see in Figure 8 that patients' waiting times are obviously shorter, hence there are fewer patients on the waiting list and the action spaces of the involved states are relatively small. Consequently, the CPU time significantly drops when λ is larger than 0.8. In addition, Figure 10(b) shows the influence of β on computational efficiency. As illustrated in Figure 8, the resulting policies for $10^{-5} \leq \beta < 1$ and $1 \leq \beta \leq 10^5$ are significantly different. Accordingly, the tendencies of \bar{t} in the two intervals are also different: \bar{t} is increasing in β for $\beta < 1$ and decreasing in β for $\beta \geq 1$.

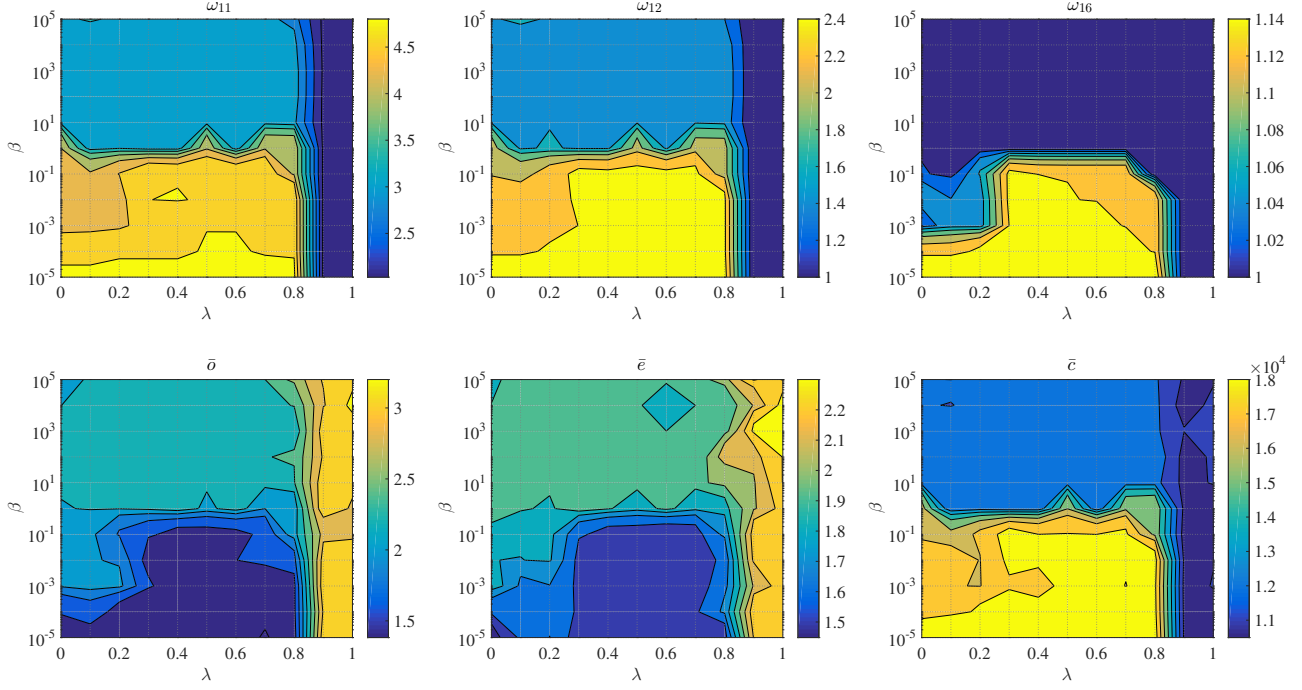


Figure 8: Experimental results for the CABG test problem with variations in λ and β

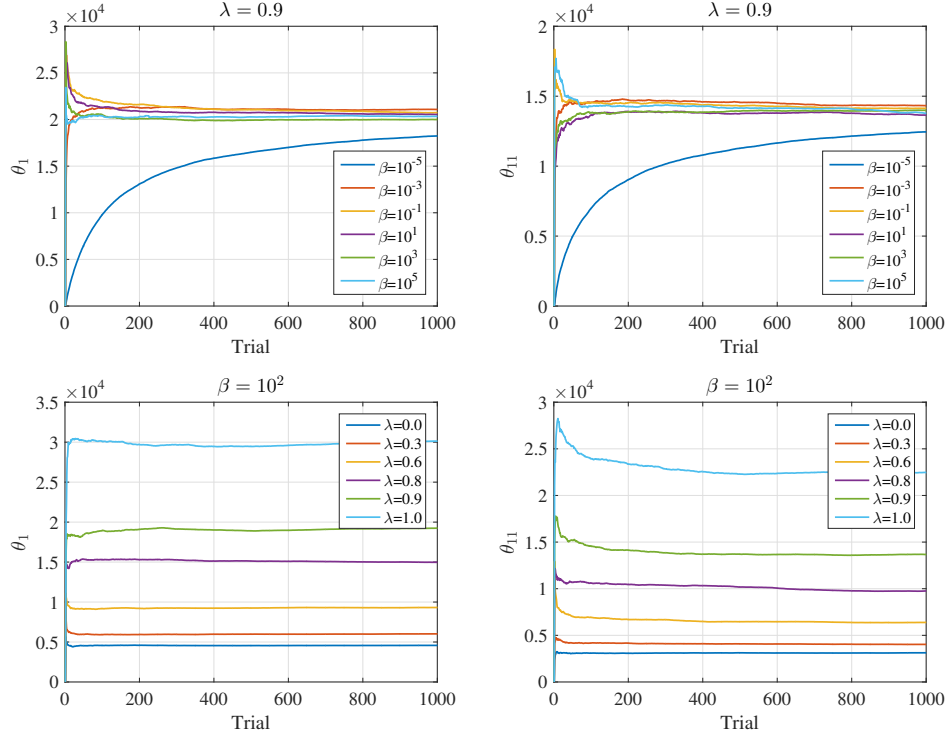


Figure 9: Evolutions of Θ when solving the CABG problem with different values of λ and β

6.3 Experimental results of a realistically sized multi-specialty patient admission control problem

In order to validate the proposed algorithms' capability to solve realistically sized problems, we consider a realistic

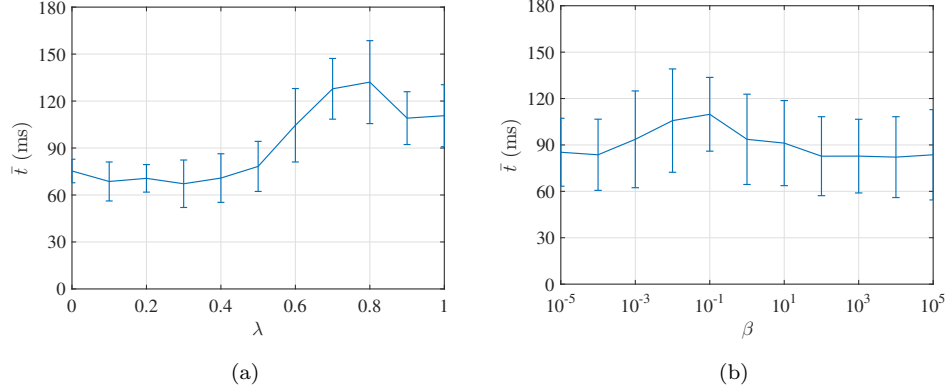


Figure 10: Comparison of CPU time consumed by the ADP* algorithm when used with different values of λ and β

multi-specialty patient admission control problem that is derived from Min and Yih (2010b) and Neyshabouri and Berg (2017) (with some reasonable modifications). The configuration of specialties is presented in Table 2. The regular OR capacity B_j listed in the last column of Table 2 is computed according to the MSS presented in Figure 1 of Min and Yih (2010b). The MSS specifies the assignments of 32 surgical blocks to 9 specialties over a one-week period. Then, given that the regular time length of each surgical block is 8 hours, the values of B_j can be computed. The unit costs are $[c_b, c_d, c_o, c_e] = [50, 200, 1000, 1000]$ and the regular SICU capacity is 105 bed-days per week (15 recovery beds). The estimated availability rates for ORs and recovery beds are $\rho_1 = \rho_2 = 0.6$. According to Table 2, we can calculate that the size of the state space is as large as 2.14×10^{176} . Given that this multi-specialty problem is on a much larger scale than the previously solved test problems, the length of the simulation is shortened to 100 weeks. ADP* is employed as the solution technique and a myopic policy is computed as the benchmark. The parameters of the ADP* algorithm are arbitrarily set as $\lambda = 0.5$, $\beta = 1$, $N = 25$, and $\epsilon = 0.01$.

Table F.3 and Table F.4 in Appendix F present the experimental results for the multi-specialty problem in detail. Table F.3 shows that, in comparison with the myopic policy ($\gamma = 0.00$), the ADP* policy ($\gamma = 0.99$) significantly shortens patients' waiting times, but leads to a slight increase in the overuse of ORs. In the specialties where the OR capacities are relatively sufficient and patients do not spend much time in the ORs and SICU, such as ENT and OPHTH, patients wait for one week at most, and there are few instances where ORs are overused. In comparison, in the specialties NEURO, CARDIAC and VASCULAR, where the OR capacities are limited and expectations of surgery duration and LOS are relatively large, many surgeries are delayed for more than one week and there is greater overuse of the ORs. From Table F.4 it can be seen that, compared to the myopic policy, ADP* reduces the total cost by two thirds, saves 9.03% of CPU time, and increases the overuse of ORs by 1.378 hours per week on average.

Figure 11 compares the evolutions of the key performance measures under the two policies (myopic and ADP*). Figure 11(a) clearly shows that the ADP* policy leads to lower costs over most of the weeks shown. Moreover, since the patients' waiting times are shortened under the ADP* policy, the costs incurred by performing and delaying surgeries (i.e., the patient-related costs) are much lower and there are significantly fewer patients on the waiting list, as shown in Figure 11(b) and 11(c). In addition, we can observe from Figure 11(d) that ADP* consumes more CPU time at the beginning of the simulation, because the parameter vector Θ is still far from convergence, whereas computing the myopic policy requires more CPU time at the end of the simulation, since the size of the waiting list and the number of actions to be evaluated continually increase over time.

Table 2: Problem configuration for the multi-specialty test problem

Specialty	j	v_j	u	W_{ju}	\bar{n}_{ju}	$\max \bar{n}_{ju}$	\bar{d}_j^*	$\sigma(d_j)^*$	\bar{l}_j^{**}	$\sigma(l_j)^{**}$	B_j^*
ENT	1	1	1	20	10.00	25	1.23	0.38	0.10	0.10	48.00
OBGYN	2	2	1	15	4.00	15	1.43	0.44	2.00	2.00	24.00
			3	6	0.50	4					
ORTHO	3	2	1	15	10.00	25	1.78	0.54	1.50	1.50	48.00
			3	6	2.00	10					
NEURO	4	5	1	8	2.50	12	2.67	1.65	2.00	2.00	8.00
GEN	5	1	1	20	9.00	20	1.55	0.67	0.05	0.05	64.00
			2	15	2.00	10					
OPHTH	6	2	1	15	1.50	8	0.63	0.10	0.05	0.05	32.00
VASCULAR	7	4	1	10	1.00	6	2.00	1.03	3.50	3.50	16.00
			2	5	2.50	12					
			4	2	0.50	4					
CARDIAC	8	5	1	8	0.25	3	4.00	2.95	2.00	2.00	8.00
			2	3	1.25	7					
			6	1	0.50	4					
UROLOGY	9	3	1	12	2.00	10	1.07	0.75	0.80	0.80	8.00
			2	6	0.50	4					

* unit: hours

** unit: days

7 Conclusion

This work deals with the admission control of elective patients. It considers uncertainty, dynamic patient priority scores, and multiple capacity constraints. Sequential decisions are made at the end of each week that determine which patients on a waiting list will be treated in the following week. To guarantee equity and efficiency, each patient on the waiting list is assigned a dynamic priority score according to his/her urgency coefficient, actual waiting time, and the relative importance of his/her specialty. The studied patient admission control problem is formulated as an infinite-horizon MDP model. The objective is to minimize a cost function that assesses the costs of performing and delaying surgeries as well as the expected costs incurred by the over-utilizations of the ORs and SICU. Given that the problem scales of real-life situations are usually very large, solving the MDP model with traditional DP algorithms is computationally intractable. Therefore, we adopt two measures to tackle the curses of dimensionality: first, we perform an intensive structural analysis for the MDP model, reducing the number of actions to be evaluated (and thus drastically reducing the dimensionality of the action space); second, we develop a novel ADP algorithm based on RLS-TD(λ) learning to tackle the dimensionalities of the state space and the outcome space.

In this paper, multiple test problems are solved in numerical experiments that validate the efficiency and the accuracy of the proposed algorithms, and sensitivity analyses for problem parameters and algorithm parameters are performed and their effects on the resulting policy and computational efficiency are evaluated. The experimental results of a small-sized test problem show that the ADP algorithm consumes significantly less CPU time and only leads to a slight increase in the total cost in comparison with conventional DP-based algorithms, even though the problem parameters vary greatly. In terms of the parameters of the ADP algorithm, our sensitivity analyses, performed on a large-sized CABG test problem, reveal that the patients' waiting time and the total costs decrease in λ and β , while the over-utilizations of ORs and SICU increase. For all the values of λ and β , the resulting ADP

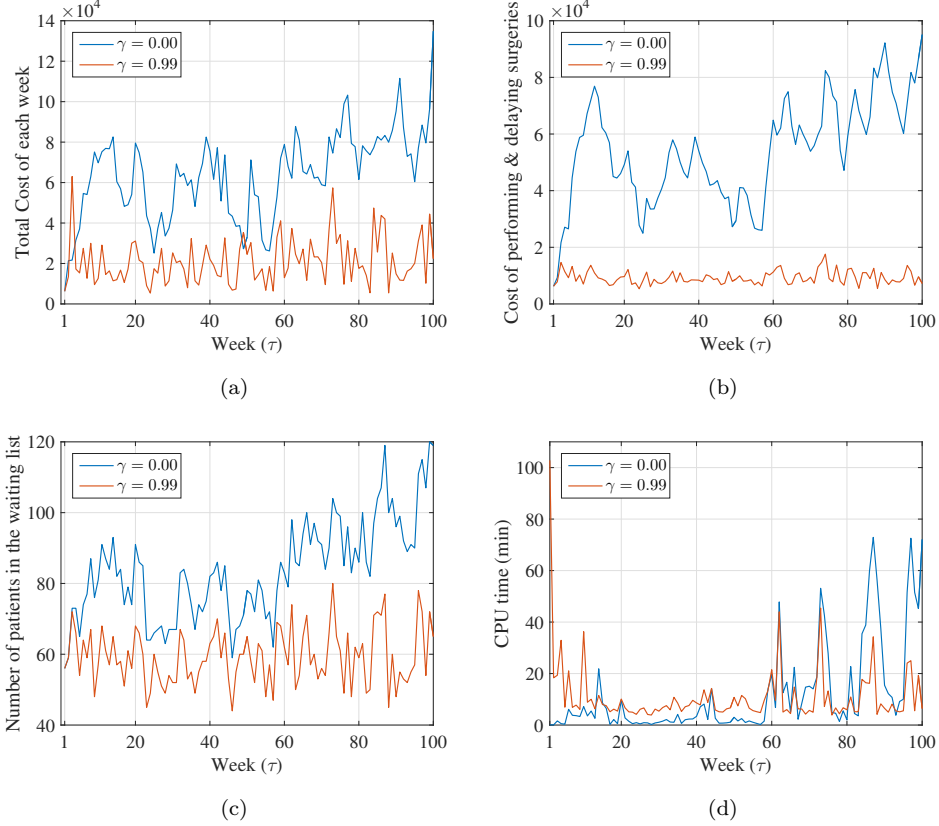


Figure 11: Comparison of two policies for the multi-specialty patient admission control problem

policy significantly outperforms the myopic policy in terms of waiting time and total cost. Following this, we solve a multi-specialty patient admission control problem using the proposed ADP algorithm and validate its capability to tackle realistically sized problems. Our experimental results also show that Algorithm 1, proposed in Section 4, drastically reduces the number of actions to be evaluated and improves the computational efficiency of all the employed algorithms.

In the future, three aspects of this work can be extended for further research. First, **stochastic surgery durations and LOSs can be explicitly incorporated into the MDP model, but** the computational burden will increase and some of the model properties proved in this paper may no longer hold true. **Therefore,** further structural analyses and more efficient solution approaches should be studied and applied to the extended MDP model. Second, we find in this work that the **RLS-TD(λ)-based ADP algorithm tends to increase the over-utilization of surgical resources and reduce patients' waiting times.** However, the exact impact of the ADP algorithm on the nature of the resulting policy is unclear, since the interactions between the MDP model and the ADP algorithm are highly complex. Hence, **an in-depth theoretical analysis of the effects of the RLS-TD(λ)-based ADP algorithm on the resulting policy can be studied as part of future research.** Third, **the MDP model formulated in this paper can easily be combined with a mathematical programming model which optimizes the intra-week surgery-to-block assignments.** For example, J. Zhang et al. (2019b) have addressed a relatively simple advance surgery scheduling problem (single specialty, single OR) using MDP + stochastic programming. Their experimental results show that the resulting surgery schedules significantly outperform those of a pure stochastic programming model. Obviously, the MDP model for patient admission control has big potential to improve the long-term quality of advance surgery scheduling, as the

costs-to-go in the future are mostly ignored in the commonly used pure mathematical programming models.

Acknowledgements

The authors gratefully acknowledge the financial support granted by the China Scholarship Council (CSC, Grant No. 201604490106).

References

- Addis, B., Carello, G., Grosso, A., & Tànfanì, E. (2016). Operating room scheduling and rescheduling: a rolling horizon approach. *Flexible Services and Manufacturing Journal*, 28(1-2), 206–232.
- Agnetis, A., Coppi, A., Corsini, M., Dellino, G., Meloni, C., & Pranzo, M. (2012). Long term evaluation of operating theater planning policies. *Operations Research for Health Care*, 1(4), 95–104.
- Agnetis, A., Coppi, A., Corsini, M., Dellino, G., Meloni, C., & Pranzo, M. (2014). A decomposition approach for the combined master surgical schedule and surgical case assignment problems. *Health care management science*, 17(1), 49–59.
- Aringhieri, R., Landa, P., Soriano, P., Tànfanì, E., & Testi, A. (2015). A two level metaheuristic for the operating room scheduling and assignment problem. *Computers & Operations Research*, 54, 21–34.
- Astaraky, D., & Patrick, J. (2015). A simulation based approximate dynamic programming approach to multi-class, multi-resource surgical scheduling. *European Journal of Operational Research*, 245(1), 309–319.
- Australian Institute of Health and Welfare. (2018). *Australia’s health 2018*. <https://www.aihw.gov.au/getmedia/7c42913d-295f-4bc9-9c24-4e44eff4a04a/aihw-aus-221.pdf.aspx?inline=true> (retrieved May 17, 2019).
- Banditori, C., Cappanera, P., & Visintin, F. (2013). A combined optimization–simulation approach to the master surgical scheduling problem. *IMA Journal of Management Mathematics*, 24(2), 155–187.
- Barto, A. G., Bradtke, S. J., & Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1-2), 81–138.
- Bonet, B., & Geffner, H. (2003). Labeled RTDP: Improving the convergence of real-time dynamic programming. In *Proceedings of Thirteenth International Conference on Automated Planning and Scheduling* (Vol. 3, pp. 12–21).
- Boyan, J. A. (2002). Technical update: Least-squares temporal difference learning. *Machine learning*, 49(2-3), 233–246.
- Centers for Medicare & Medicaid Services. (2018). *National health expenditures 2017 highlights*. <https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/NationalHealthExpendData/Downloads/highlights.pdf> (retrieved May 17, 2019).
- Denton, B., Miller, A., Balasubramanian, H., & Huschka, T. (2010). Optimal allocation of surgery blocks to operating rooms under uncertainty. *Operations research*, 58(4-part-1), 802–816.
- Denton, B., Viapiano, J., & Vogl, A. (2007). Optimization of surgery sequencing and scheduling decisions under uncertainty. *Health care management science*, 10(1), 13–24.
- Dios, M., Molina-Pariente, J. M., Fernandez-Viagas, V., Andrade-Pineda, J. L., & Framinan, J. M. (2015). A decision support system for operating room scheduling. *Computers & Industrial Engineering*, 88, 430–443.
- Duma, D., & Aringhieri, R. (2019). The management of non-elective patients: shared vs. dedicated policies. *Omega*, 83, 199–212.
- Fei, H., Chu, C., & Meskens, N. (2009). Solving a tactical operating room planning problem by a column-generation-based heuristic procedure with four criteria. *Annals of Operations Research*, 166(1), 91.

- Fei, H., Chu, C., Meskens, N., & Artiba, A. (2008). Solving surgical cases assignment problem by a branch-and-price approach. *International Journal of Production Economics*, 112(1), 96–108.
- Fei, H., Meskens, N., & Chu, C. (2010). A planning and scheduling problem for an operating theatre using an open scheduling strategy. *Computers & Industrial Engineering*, 58(2), 221–230.
- Fei, H., Meskens, N., Combes, C., & Chu, C. (2009). The endoscopy scheduling problem: A case study with two specialised operating rooms. *International Journal of Production Economics*, 120(2), 452–462.
- Ferrand, Y., Magazine, M., & Rao, U. (2010). Comparing two operating-room-allocation policies for elective and emergency surgeries. In *Proceedings of the 2010 Winter Simulation Conference* (pp. 2364–2374).
- Ferrand, Y., Magazine, M., & Rao, U. (2014). Partially flexible operating rooms for elective and emergency surgeries. *Decision Sciences*, 45(5), 819–847.
- Gerchak, Y., Gupta, D., & Henig, M. (1996). Reservation planning for elective surgery under uncertain demand for emergency surgery. *Management Science*, 42(3), 321–334.
- Gocgun, Y., & Ghathe, A. (2012). Lagrangian relaxation and constraint generation for allocation and advanced scheduling. *Computers & Operations Research*, 39(10), 2323–2336.
- Green, L. V., Savin, S., & Wang, B. (2006). Managing patient service in a diagnostic medical facility. *Operations Research*, 54(1), 11–25.
- Guerriero, F., & Guido, R. (2011). Operational research in the management of the operating theatre: A survey. *Health Care Management Science*, 14(1), 89–114.
- Guido, R., & Conforti, D. (2017). A hybrid genetic approach for solving an integrated multi-objective operating room planning and scheduling problem. *Computers & Operations Research*, 87, 270–282.
- Hashemi Doulabi, S. H., Rousseau, L.-M., & Pesant, G. (2016). A constraint-programming-based branch-and-price-and-cut approach for operating room planning and scheduling. *INFORMS Journal on Computing*, 28(3), 432–448.
- Hosseini, N., & Taaffe, K. (2014). Evaluation of optimal scheduling policy for accommodating elective and non-elective surgery via simulation. In *Proceedings of the 2014 Winter Simulation Conference* (pp. 1377–1386).
- Huh, W. T., Liu, N., & Truong, V.-A. (2013). Multiresource allocation scheduling in dynamic environments. *Manufacturing & Service Operations Management*, 15(2), 280–291.
- Jebali, A., & Diabat, A. (2015). A stochastic model for operating room planning under capacity constraints. *International Journal of Production Research*, 53(24), 7252–7270.
- Jebali, A., & Diabat, A. (2017). A chance-constrained operating room planning with elective and emergency cases under downstream capacity constraints. *Computers & Industrial Engineering*, 114, 329–344.
- Jonnalagadda, R., Walrond, E., Hariharan, S., Walrond, M., & Prasad, C. (2005). Evaluation of the reasons for cancellations and delays of surgical procedures in a developing country. *International journal of clinical practice*, 59(6), 716–720.
- Koppka, L., Wiesche, L., Schacht, M., & Werners, B. (2018). Optimal distribution of operating hours over operating rooms using probabilities. *European Journal of Operational Research*, 267(3), 1156–1171.
- Lamiri, M., Xie, X., Dolgui, A., & Grimaud, F. (2008). A stochastic model for operating room planning with elective and emergency demand for surgery. *European Journal of Operational Research*, 185(3), 1026–1037.
- Lamiri, M., Xie, X., & Zhang, S. (2008). Column generation approach to operating theater planning with elective and emergency patients. *IIE Transactions*, 40(9), 838–852.
- Liu, N., Ziya, S., & Kulkarni, V. G. (2010). Dynamic scheduling of outpatient appointments under patient no-shows and cancellations. *Manufacturing & Service Operations Management*, 12(2), 347–364.
- Ljung, L., & Söderström, T. (1983). *Theory and practice of recursive identification*. Cambridge, MA: MIT Press.

- Marques, I., & Captivo, M. E. (2017). Different stakeholders' perspectives for a surgical case assignment problem: Deterministic and robust approaches. *European Journal of Operational Research*, 261(1), 260–278.
- Mausam, & Kolobov, A. (2012). Planning with Markov decision processes: An AI perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1), 1–210.
- McMahan, H. B., Likhachev, M., & Gordon, G. J. (2005). Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In *Proceedings of the 22nd International Conference on Machine Learning* (pp. 569–576).
- Min, D., & Yih, Y. (2010a). An elective surgery scheduling problem considering patient priority. *Computers & Operations Research*, 37(6), 1091–1099.
- Min, D., & Yih, Y. (2010b). Scheduling elective surgery under uncertainty and downstream capacity constraints. *European Journal of Operational Research*, 206(3), 642–652.
- Min, D., & Yih, Y. (2014). Managing a patient waiting list with time-dependent priority and adverse events. *RAIRO-Operations Research*, 48(1), 53–74.
- Molina-Pariente, J. M., Fernandez-Viagas, V., & Framinan, J. M. (2015). Integrated operating room planning and scheduling problem with assistant surgeon dependent surgery durations. *Computers & Industrial Engineering*, 82, 8–20.
- Molina-Pariente, J. M., Hans, E. W., Framinan, J. M., & Gomez-Cia, T. (2015). New heuristics for planning operating rooms. *Computers & Industrial Engineering*, 90, 429–443.
- Moosavi, A., & Ebrahimnejad, S. (2018). Scheduling of elective patients considering upstream and downstream units and emergency demand using robust optimization. *Computers & Industrial Engineering*, 120, 216–233.
- National Bureau of Statistics of China. (2019). *China statistical yearbook 2018*. <http://www.stats.gov.cn/tjsj/ndsj/2018/indexeh.htm> (retrieved May 17, 2019).
- Neyshabouri, S., & Berg, B. P. (2017). Two-stage robust optimization approach to elective surgery and downstream capacity planning. *European Journal of Operational Research*, 260(1), 21–40.
- Pang, B., Xie, X., Song, Y., & Luo, L. (2018). Surgery scheduling under case cancellation and surgery duration uncertainty. *IEEE Transactions on Automation Science and Engineering*, 16(1), 74–86.
- Patrick, J., Puterman, M. L., & Queyranne, M. (2008). Dynamic multipriority patient scheduling for a diagnostic resource. *Operations research*, 56(6), 1507–1525.
- Powell, W. B. (2011). *Approximate dynamic programming: Solving the curses of dimensionality* (2nd ed.). Hoboken, New Jersey: John Wiley & Sons, Inc.
- Powell, W. B., & Meisel, S. (2015). Tutorial on stochastic optimization in energy—Part II: An energy storage illustration. *IEEE Transactions on Power Systems*, 31(2), 1468–1475.
- Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. New York: John Wiley & Sons, Inc.
- Rachuba, S., & Werners, B. (2017). A fuzzy multi-criteria approach for robust operating room schedules. *Annals of Operations Research*, 251(1-2), 325–350.
- Rath, S., Rajaram, K., & Mahajan, A. (2017). Integrated anesthesiologist and room scheduling for surgeries: Methodology and application. *Operations Research*, 65(6), 1460–1478.
- Roshanaei, V., Luong, C., Aleman, D. M., & Urbach, D. (2017b). Propagating logic-based benders' decomposition approaches for distributed operating room scheduling. *European Journal of Operational Research*, 257(2), 439–455.
- Roshanaei, V., Luong, C., Aleman, D. M., & Urbach, D. R. (2017a). Collaborative operating room planning and scheduling. *INFORMS Journal on Computing*, 29(3), 558–580.

- Roshanaei, V., Luong, C., Aleman, D. M., & Urbach, D. R. (2020). Reformulation, linearization, and decomposition techniques for balanced distributed operating room scheduling. *Omega*, 93, 102043.
- Samudra, M., Van Riet, C., Demeulemeester, E., Cardoen, B., Vansteenkiste, N., & Rademakers, F. E. (2016). Scheduling operating rooms: Achievements, challenges and pitfalls. *Journal of Scheduling*, 19(5), 493–525.
- Sanner, S., Goetschalckx, R., Driessens, K., & Shani, G. (2009). Bayesian real-time dynamic programming. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence* (pp. 1784–1789).
- Smith, T., & Simmons, R. (2006). Focused real-time dynamic programming for MDPs: Squeezing more out of a heuristic. In *Proceedings of the 21st National Conference on Artificial Intelligence* (Vol. 2, pp. 1227–1232).
- Sobolev, B., & Kuramoto, L. (2008). *Analysis of waiting-time data in health services research*. New York: Springer Science & Business Media.
- Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, 3(1), 9–44.
- Tancrez, J.-S., Roland, B., Cordier, J.-P., & Riane, F. (2013). Assessing the impact of stochasticity for operating theater sizing. *Decision Support Systems*, 55(2), 616–628.
- Tànfanì, E., & Testi, A. (2010). A pre-assignment heuristic algorithm for the master surgical schedule problem (mssp). *Annals of Operations Research*, 178(1), 105–119.
- Testi, A., & Tànfani, E. (2009). Tactical and operational decisions for operating room planning: Efficiency and welfare implications. *Health Care Management Science*, 12(4), 363.
- Testi, A., Tanfani, E., & Torre, G. (2007). A three-phase approach for operating theatre schedules. *Health Care Management Science*, 10(2), 163–172.
- Testi, A., Tanfani, E., Valente, R., Ansaldo, G., & Torre, G. (2008). Prioritizing surgical waiting lists. *Journal of Evaluation in Clinical Practice*, 14(1), 59–64.
- Truong, V.-A. (2015). Optimal advance scheduling. *Management Science*, 61(7), 1584–1597.
- Tsitsiklis, J. N., & Van Roy, B. (1997, May). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5), 674–690. doi: 10.1109/9.580874
- Ulmer, M. W., & Thomas, B. W. (2020). Meso-parametric value function approximation for dynamic customer acceptances in delivery routing. *European Journal of Operational Research*, 285(1), 183–195.
- Utzolino, S., Kaffarnik, M., Keck, T., Berlet, M., & Hopt, U. T. (2010). Unplanned discharges from a surgical intensive care unit: readmissions and mortality. *Journal of critical care*, 25(3), 375–381.
- Valente, R., Testi, A., Tanfani, E., Fato, M., Porro, I., Santo, M., ... Ansaldo, G. (2009). A model to prioritize access to elective surgery on the basis of clinical urgency and waiting time. *BMC Health Services Research*, 9(1), 1.
- Vancroonenburg, W., De Causmaecker, P., & Berghe, G. V. (2019). Chance-constrained admission scheduling of elective surgical patients in a dynamic, uncertain setting. *Operations Research for Health Care*, 22, 100196.
- Van Riet, C., & Demeulemeester, E. (2015). Trade-offs in operating room planning for electives and emergencies: A review. *Operations Research for Health Care*, 7, 52–69.
- Voelkel, M. A., Sachs, A.-L., & Thonemann, U. W. (2020). An aggregation-based approximate dynamic programming approach for the periodic review model with random yield. *European Journal of Operational Research*, 281(2), 286–298.
- Wang, S., Roshanaei, V., Aleman, D., & Urbach, D. (2016). A discrete event simulation evaluation of distributed operating room scheduling. *IIE Transactions on Healthcare Systems Engineering*, 6(4), 236–245.
- Wang, Y., Tang, J., & Fung, R. Y. (2014). A column-generation-based heuristic algorithm for solving operating theater planning problem under stochastic demand and surgery cancellation risk. *International Journal of Production Economics*, 158, 28–36.

- White, C. C. (2001). Markov decision processes. In *Encyclopedia of operations research and management science* (pp. 484–486). Boston, MA: Springer US.
- Xiao, G., van Jaarsveld, W., Dong, M., & Van De Klundert, J. (2016). Stochastic programming analysis and solutions to schedule overcrowded operating rooms in china. *Computers & Operations Research*, 74, 78–91.
- Xu, X., He, H.-g., & Hu, D. (2002). Efficient reinforcement learning using recursive least-squares methods. *Journal of Artificial Intelligence Research*, 16, 259–292.
- Yin, B., Dridi, M., & Moudni, A. E. (2017, June). Recursive least-squares temporal difference learning for adaptive traffic signal control at intersection. *Neural Computing and Applications*, 1–16.
- Zhang, J., Dridi, M., & El Moudni, A. (2019a). A markov decision model with dead ends for operating room planning considering dynamic patient priority. *RAIRO-Operations Research*, 53(5), 1819–1841.
- Zhang, J., Dridi, M., & El Moudni, A. (2019b). A two-level optimization model for elective surgery scheduling with downstream capacity constraints. *European Journal of Operational Research*, 276(2), 602–613.
- Zhang, J., Dridi, M., & El Moudni, A. (2020). Column-generation-based heuristic approaches to stochastic surgery scheduling with downstream capacity constraints. *International Journal of Production Economics*, 107764.
- Zhang, Y., Wang, Y., Tang, J., & Lim, A. (2020). Mitigating overtime risk in tactical surgical scheduling. *Omega*, 93, 102024.
- Zhu, S., Fan, W., Yang, S., Pei, J., & Pardalos, P. M. (2019). Operating room planning and surgical case scheduling: a review of literature. *Journal of Combinatorial Optimization*, 37(3), 757–805.

Appendix A Proof of Lemma 1

Proof. Let $x_1 = \arg \min f(x)$ and $x_2 = \arg \min g(x)$, then $g(x_1) \geq g(x_2)$, hence

$$\min f(x) - \min g(x) = f(x_1) - g(x_2) \geq f(x_1) - g(x_1) \geq \min[f(x) - g(x)] \quad (\text{A.1})$$

thus the lemma is proved. \square

Appendix B Proof of Proposition 1

Proof. (i) If $a_0(s + \Delta_{j'u'w'}) \in A(s)$, then by Lemma 1,

$$\begin{aligned} C_0(s + \Delta_{j'u'w'}) - C_0(s) &= \min_{a \in A(s)} C(s + \Delta_{j'u'w'}, a) - \min_{a \in A(s)} C(s, a) \\ &\geq \min_{a \in A(s)} [C(s + \Delta_{j'u'w'}, a) - C(s, a)] = c_d v_{j'u'w'} > 0 \end{aligned} \quad (\text{B.1})$$

Otherwise, $a_0(s + \Delta_{j'u'w'}) \in A(s + \Delta_{j'u'w'}) \setminus A(s)$. Let $s = \{n_{j'uw}\}$, $a_0(s) = \{m_{j'uw}^0\}$ and $a_0(s + \Delta_{j'u'w'}) = \{m_{j'uw}'\}$, then we have $m_{j'uw}' = n_{j'u'w'} + 1$ and $a_0(s + \Delta_{j'u'w'}) - \Delta_{j'u'w'} \in A(s)$. Since $a_0(s)$ minimizes $C(s, a)$, then

$$\begin{aligned} &C_0(s) - C[s, a_0(s + \Delta_{j'u'w'}) - \Delta_{j'u'w'}] \\ &= C_0(s) - C_p[s, a_0(s + \Delta_{j'u'w'})] + (c_b - c_d) v_{j'u'w'} - C_h[a_0(s + \Delta_{j'u'w'}) - \Delta_{j'u'w'}] \leq 0 \end{aligned} \quad (\text{B.2})$$

As a result,

$$C_p[s, a_0(s + \Delta_{j'u'w'})] - C_0(s) \geq (c_b - c_d) v_{j'u'w'} - C_h[a_0(s + \Delta_{j'u'w'}) - \Delta_{j'u'w'}] \quad (\text{B.3})$$

Given that $C_h[a_0(s + \Delta_{j'u'w'})] \geq C_h[a_0(s + \Delta_{j'u'w'}) - \Delta_{j'u'w'}]$, we have

$$\begin{aligned}
& C_0(s + \Delta_{j'u'w'}) - C_0(s) \\
&= C_p[s + \Delta_{j'u'w'}, a_0(s + \Delta_{j'u'w'})] + C_h[a_0(s + \Delta_{j'u'w'})] - C_0(s) \\
&= C_p[s, a_0(s + \Delta_{j'u'w'})] + c_d v_{j'} u' w' + C_h[a_0(s + \Delta_{j'u'w'})] - C_0(s) \\
&\geq c_b v_{j'} u' w' + C_h[a_0(s + \Delta_{j'u'w'})] - C_h[a_0(s + \Delta_{j'u'w'}) - \Delta_{j'u'w'}] > 0
\end{aligned} \tag{B.4}$$

As $C_0(s + \Delta_{j'u'w'}) - C_0(s) > 0$ holds in all the possible cases, $C_0(s)$ is increasing in s .

- (ii) By Definition 2, if $a = \{m_{juw}\}$ and $a' = \{m'_{juw}\}$ are comparable, then $\sum_{u=1}^{U_j} \sum_{w=1}^{W_{ju}} m_{juw} = \sum_{u=1}^{U_j} \sum_{w=1}^{W_{ju}} m'_{juw}$ holds for any specialty j . Further, according to the model descriptions presented in Section 3, patients from the same specialty have the same expectations of surgery duration and LOS. Hence by the cost function (4), $C_h(a) = C_h(a')$.
- (iii) Let $\sigma = s - \Delta_{j'u'w'} = s' - \Delta_{j'u''w''}$ and $u'w' < u''w''$, then $P(s) < P(s')$. Let $\sigma = \{\sigma_{juw}\}$, $a_0(s) = \{m_{juw}^s\}$ and $a_0(s') = \{m_{juw}^{s'}\}$, then we discuss all the possible cases:
 If $a_0(s') \in A(s') \setminus A(\sigma)$, then $m_{j'u''w''}^{s'} = \sigma_{j'u''w''} + 1$ and $m_{j'u'w'}^{s'} \leq \sigma_{j'u'w'} + 1$, thus $a' = a_0(s') - \Delta_{j'u''w''} + \Delta_{j'u'w'} \in A(s)$. Since $C_h(a') = C_h[a_0(s')]$ by (ii) and $C_0(s) = \min_{a \in A(s)} C(s, a)$, then

$$\begin{aligned}
& C_0(s') - C_0(s) = C_p[s', a_0(s')] + C_h[a_0(s')] - C_0(s) \\
&= C_p(s, a') + c_b v_{j'}(u''w'' - u'w') + C_h(a') - C_0(s) > C(s, a') - C_0(s) \geq 0
\end{aligned} \tag{B.5}$$

Otherwise, $a_0(s') \in A(\sigma)$. We suppose that $a_0(s) \in A(s) \setminus A(\sigma)$, then $m_{j'u'w'}^s = \sigma_{j'u'w'} + 1$ and $m_{j'u''w''}^s \leq \sigma_{j'u''w''} + 1$, thus $a' = a_0(s) - \Delta_{j'u'w'} + \Delta_{j'u''w''} \in A(s')$. Since $C_0(s') = \min_{a \in A(s')} C(s', a)$, and $C_h(a') = C_h[a_0(s)]$ by (i), then

$$\begin{aligned}
& C[s, a_0(s)] - C[s, a_0(s')] = C(s', a') + (c_b - c_d) v_{j'}(u'w' - u''w'') - C[s', a_0(s')] \\
&> C(s', a') - C_0(s') \geq 0
\end{aligned} \tag{B.6}$$

As $C[s, a_0(s)] - C[s, a_0(s')] > 0$ contradicts the fact that $a_0(s)$ minimizes $C(s, a)$, $a_0(s) \in A(s) \setminus A(\sigma)$ does not hold, hence $a_0(s) \in A(\sigma)$ holds when $a_0(s') \in A(\sigma)$. Then by Lemma 1, we have

$$C_0(s') - C_0(s) \geq \min_{a \in A(\sigma)} [C(s', a) - C(s, a)] = c_d v_{j'}(u''w'' - u'w') > 0 \tag{B.7}$$

To summarize, since $C_0(s') - C_0(s) > 0$ holds in all the possible cases, then $C_0(s)$ is increasing in $P(s)$. \square

Appendix C Proof of Proposition 2

Proof. (i) Since $V_1^{\pi^*}(s) = V^{\pi^*}(s)$, we first prove that $V_n^{\pi^*}(s)$ is increasing in s for $n = 1, 2, \dots, N$. The proof is given by backward mathematical induction:

For $n = N$, since $V_{N+1}^{\pi^*}(s') = 0$ holds for any $s' \in S$, then $V_N^{\pi^*}(s) = \min_{a \in A(s)} C(s, a) = C_0(s)$. $C_0(s)$ is increasing in s by (i) of Proposition 1, so is $V_N^{\pi^*}(s)$.

For $n = k < N$, suppose that $V_{k+1}^{\pi^*}(s)$ is increasing in s , then we distinguish the following cases:

If $\pi^*(s + \Delta_{j'u'w'}) \in A(s)$, then by Lemma 1,

$$\begin{aligned}
& V_k^{\pi^*}(s + \Delta_{j'u'w'}) - V_k^{\pi^*}(s) \geq \min_{a \in A(s)} [Q_k^{\pi^*}(s + \Delta_{j'u'w'}, a) - Q_k^{\pi^*}(s, a)] \\
&= \min_{a \in A(s)} \left\{ c_d v_{j'} u' w' + \gamma \sum_{\Psi=0}^{+\infty} P_\Psi[V_{k+1}^{\pi^*}(G_a^{s+\Delta_{j'u'w'}} + \Psi) - V_{k+1}^{\pi^*}(G_a^s + \Psi)] \right\} > 0
\end{aligned} \tag{C.1}$$

Otherwise, $\pi^*(s + \Delta_{j'u'w'}) \in A(s + \Delta_{j'u'w'}) \setminus A(s)$. Let $s = \{n_{j'uw}\}$, $\pi^*(s) = \{m_{j'uw}^*\}$ and $\pi^*(s + \Delta_{j'u'w'}) = \{m_{j'uw}^+\}$, then we have $m_{j'u'w'}^+ = n_{j'u'w'} + 1$ and $\pi^- = \pi^*(s + \Delta_{j'u'w'}) - \Delta_{j'u'w'} \in A(s)$, thus

$$\begin{aligned}
& V_k^{\pi^*}(s + \Delta_{j'u'w'}) - V_k^{\pi^*}(s) \\
&= C[s + \Delta_{j'u'w'}, \pi^- + \Delta_{j'u'w'}] + \gamma \sum_{\Psi=0}^{+\infty} P_\Psi V_{k+1}^{\pi^*}(G_{\pi^- + \Delta_{j'u'w'}}^{s + \Delta_{j'u'w'}} + \Psi) - V_k^{\pi^*}(s) \\
&= C[s, \pi^-] + c_b v_{j'} u' w' + C_h(\pi^- + \Delta_{j'u'w'}) - C_h(\pi^-) + \gamma \sum_{\Psi=0}^{+\infty} P_\Psi V_{k+1}^{\pi^*}(G_{\pi^-}^s + \Psi) - V_k^{\pi^*}(s) \\
&> C[s, \pi^-] + \gamma \sum_{\Psi=0}^{+\infty} P_\Psi V_{k+1}^{\pi^*}(G_{\pi^-}^s + \Psi) - V_k^{\pi^*}(s) = Q_k^{\pi^*}(s, \pi^-) - V_k^{\pi^*}(s) \geq 0
\end{aligned} \tag{C.2}$$

As $V_k^{\pi^*}(s + \Delta_{j'u'w'}) > V_k^{\pi^*}(s)$ holds in all the two possible cases, the induction hypothesis is satisfied, thus $V_n^{\pi^*}(s)$ is increasing in s for $n = 1, 2, \dots, N$. Therefore, $V^{\pi^*}(s) = V_1^{\pi^*}(s)$ is increasing in s .

(ii) We employ backward mathematical induction to prove that $V_n^{\pi^*}(s)$ is increasing in $P(s)$ for $n = 1, 2, \dots, N$:

For $n = N$, since $\forall s' \in S$: $V_{N+1}^{\pi^*}(s') = 0$, then $V_N^{\pi^*}(s) = \min_{a \in A(s)} C(s, a) = C_0(s)$. $C_0(s)$ is increasing in $P(s)$ by (iii) of Proposition 1, so is $V_N^{\pi^*}(s)$.

For $n = k < N$, suppose that $V_{k+1}^{\pi^*}(s)$ is increasing in $P(s)$. Let $\sigma = s - \Delta_{j'u'w'} = s' - \Delta_{j'u''w''}$ and $u'w' < u''w''$, then $P(s) < P(s')$. Similar to the proof of (iii) of Proposition 1, the following cases are considered:

If $\pi^*(s') \in A(s') \setminus A(\sigma)$, then $a' = \pi^*(s') - \Delta_{j'u''w''} + \Delta_{j'u'w'} \in A(s)$, and $C_h(a') = C_h[\pi^*(s')]$ by (ii) of Proposition 1, hence

$$\begin{aligned}
& V_k^{\pi^*}(s') - V_k^{\pi^*}(s) = C[s', \pi^*(s')] + \gamma \sum_{\Psi=0}^{+\infty} P_\Psi V_{k+1}^{\pi^*}(G_{\pi^*(s')}^{s'} + \Psi) - V_k^{\pi^*}(s) \\
&= C(s, a') + c_b v_{j'}(u''w'' - u'w') + \gamma \sum_{\Psi=0}^{+\infty} P_\Psi V_{k+1}^{\pi^*}(G_{a'}^s + \Psi) - V_k^{\pi^*}(s) > Q_k^{\pi^*}(s, a') - V_k^{\pi^*}(s) \geq 0
\end{aligned} \tag{C.3}$$

Otherwise, $\pi^*(s') \in A(\sigma)$. Suppose that $\pi^*(s) \in A(s) \setminus A(\sigma)$, then $a' = \pi^*(s) - \Delta_{j'u'w'} + \Delta_{j'u''w''} \in A(s')$. Because $P(G_{\pi^*(s')}^{s'} + \Psi) < P(G_{\pi^*(s)}^s + \Psi)$, then $V_{k+1}^{\pi^*}(G_{\pi^*(s')}^{s'} + \Psi) < V_{k+1}^{\pi^*}(G_{\pi^*(s)}^s + \Psi)$. Given that $C_h[\pi^*(s)] = C_h(a')$ by (ii) of Proposition 1, we have

$$\begin{aligned}
& V_k^{\pi^*}(s) - Q_k^{\pi^*}[s, \pi^*(s')] = C[s, \pi^*(s)] - C[s, \pi^*(s')] + \gamma \sum_{\Psi=0}^{+\infty} P_\Psi [V_{k+1}^{\pi^*}(G_{\pi^*(s)}^s + \Psi) - V_{k+1}^{\pi^*}(G_{\pi^*(s')}^{s'} + \Psi)] \\
&> C(s', a') - C[s', \pi^*(s')] + (c_b - c_d) v_{j'}(u'w' - u''w'') + \gamma \sum_{\Psi=0}^{+\infty} P_\Psi [V_{k+1}^{\pi^*}(G_{a'}^{s'} + \Psi) - V_{k+1}^{\pi^*}(G_{\pi^*(s')}^{s'} + \Psi)] \\
&> Q_k^{\pi^*}(s', a') - V_k^{\pi^*}(s') \geq 0
\end{aligned} \tag{C.4}$$

As $V_k^{\pi^*}(s) - Q_k^{\pi^*}[s, \pi^*(s')] > 0$ contradicts the fact that $\pi^*(s)$ minimizes $Q_k^{\pi^*}(s, a)$, $\pi^*(s) \in A(s) \setminus A(\sigma)$ does not hold, hence $\pi^*(s) \in A(\sigma)$. Then by Lemma 1,

$$\begin{aligned}
& V_k^{\pi^*}(s') - V_k^{\pi^*}(s) \geq \min_{a \in A(\sigma)} [Q_k^{\pi^*}(s', a) - Q_k^{\pi^*}(s, a)] \\
&= \min_{a \in A(\sigma)} \left\{ c_d v_{j'}(u''w'' - u'w') + \gamma \sum_{\Psi=0}^{+\infty} P_\Psi [V_{k+1}^{\pi^*}(G_a^{s'} + \Psi) - V_{k+1}^{\pi^*}(G_a^s + \Psi)] \right\} > 0
\end{aligned} \tag{C.5}$$

Since $V_k^{\pi^*}(s') - V_k^{\pi^*}(s) > 0$ holds in all the possible cases, the induction hypothesis is satisfied, thus $V_n^{\pi^*}(s)$ is increasing in $P(s)$ for $n = 1, 2, \dots, N$, hence $V^{\pi^*}(s) = V_1^{\pi^*}(s)$ is increasing in $P(s)$. \square

Appendix D Proof of Proposition 3

Proof. Let $s = \{n_{j'uw'}\}$, $\pi^*(s) = \{m_{j'uw'}^*\}$ and $(c_d - c_b)v_{j'}u'w' > c_o\bar{d}_{j'} + c_e\bar{l}_{j'}$. Suppose that $n_{j'uw'} - m_{j'uw'}^* = x > 0$, then $a' = \pi^*(s) + x\Delta_{j'uw'} \in A(s)$. By (4), we know that $C_h(a') - C_h[\pi^*(s)] \leq (c_o\bar{d}_{j'} + c_e\bar{l}_{j'})x$, then

$$\begin{aligned} C[s, \pi^*(s)] - C(s, a') &= (c_d - c_b)v_{j'}u'w'x + C_h[\pi^*(s)] - C_h(a') \\ &\geq [(c_d - c_b)v_{j'}u'w' - c_o\bar{d}_{j'} - c_e\bar{l}_{j'}]x > 0 \end{aligned} \quad (\text{D.1})$$

Besides, since $G_{\pi^*(s)}^s + \Psi > G_{a'}^s + \Psi$, and $V^{\pi^*}(s)$ is increasing in s by (i) of Proposition 2, then $V^{\pi^*}(G_{\pi^*(s)}^s + \Psi) - V^{\pi^*}(G_{a'}^s + \Psi) > 0$. Therefore,

$$Q^{\pi^*}[s, \pi^*(s)] - Q^{\pi^*}(s, a') = C[s, \pi^*(s)] - C(s, a') + \gamma \sum_{\Psi=0}^{+\infty} P_{\Psi}[V^{\pi^*}(G_{\pi^*(s)}^s + \Psi) - V^{\pi^*}(G_{a'}^s + \Psi)] > 0 \quad (\text{D.2})$$

Since $Q^{\pi^*}[s, \pi^*(s)] - Q^{\pi^*}(s, a') > 0$ contradicts the fact that $\pi^*(s)$ minimizes $Q^{\pi^*}(s, a)$, then $n_{j'uw'} - m_{j'uw'}^* = x > 0$ does not hold, proving the proposition. \square

Appendix E Proof of Proposition 4

Proof. Let $a = a' + \Delta_{j'uw'} - \Delta_{j'u''w''}$ and $u'w' < u''w''$, then $P(a) < P(a')$ and $P(G_a^s + \Psi) > P(G_{a'}^s + \Psi)$, thus $V^{\pi^*}(G_a^s + \Psi) > V^{\pi^*}(G_{a'}^s + \Psi)$ holds by (ii) of Proposition 2, hence

$$Q^{\pi^*}(s, a) - Q^{\pi^*}(s, a') = (c_b - c_d)v_{j'}(u'w' - u''w'') + \gamma \sum_{\Psi=0}^{+\infty} P_{\Psi}[V^{\pi^*}(G_a^s + \Psi) - V^{\pi^*}(G_{a'}^s + \Psi)] > 0 \quad (\text{E.1})$$

As $Q^{\pi^*}(s, a) - Q^{\pi^*}(s, a') > 0$ holds for $P(a) < P(a')$, the proposition is proved. \square

Appendix F Detailed experimental results of Section 6

Table F.1: Experimental results of the small-sized test problem

Algorithm	Type	$\bar{\omega}_{11}$	$\sigma(\omega_{11})$	$\bar{\omega}_{12}$	$\sigma(\omega_{12})$	$\bar{\omega}_{21}$	$\sigma(\omega_{21})$	$\bar{\omega}_{22}$	$\sigma(\omega_{22})$	\bar{o}_1	$\sigma(o_1)$	\bar{o}_2	$\sigma(o_2)$	\bar{e}	$\sigma(e)$	\bar{c}	$\sigma(c)$	T	$\frac{\ A^*\ }{\ A\ }$
VI	off-line	2.293	0.938	1.293	0.455	1.463	0.599	1.051	0.219	0.965	2.081	1.162	2.970	2.341	5.222	3.716	5,669	85,492	1.000
VI*	off-line	2.280	0.922	1.304	0.460	1.474	0.606	1.051	0.219	0.961	2.099	1.160	2.971	2.340	5.234	3.711	5,706	6,835	0.063
VPI-RTDP	on-line	2.277	0.925	1.316	0.465	1.452	0.604	1.046	0.210	0.957	2.101	1.161	2.973	2.336	5.226	3,704	5,701	5,437	1.000
VPI-RTDP*	on-line	2.277	0.925	1.316	0.465	1.452	0.604	1.046	0.210	0.957	2.101	1.161	2.973	2.336	5.226	3,704	5,701	1,450	0.216
ADP	on-line	2.521	1.090	1.432	0.495	1.099	0.299	1.000	0.000	0.997	2.168	1.186	3.037	2.373	5.383	3,821	5,808	14	1.000
ADP*	on-line	2.521	1.092	1.432	0.495	1.096	0.294	1.000	0.000	0.998	2.166	1.186	3.026	2.371	5.384	3,820	5,802	12	0.901

Algorithms with superscript * evaluate the action set $A^*(s)$ determined by Algorithm 1, the others enumerate the entire action space $A^*(s) = A(s)$.

Table F.2: Experimental results for the CABG problem

γ	λ	β	$\bar{\omega}_{11}$	$\sigma(\omega_{11})$	$\bar{\omega}_{12}$	$\sigma(\omega_{12})$	$\bar{\omega}_{16}$	$\sigma(\omega_{16})$	\bar{o}	$\sigma(o)$	\bar{e}	$\sigma(e)$	\bar{c}	$\sigma(c)$	\bar{t}	$\sigma(t)$	$\frac{\ A^*\ }{\ A\ }$ (%)
0.00	—	—	4.855	2.362	2.493	1.168	1.159	0.366	1.658	2.583	1.697	2.808	19008.242	12651.372	0.011	0.104	<0.001
0.99	0.0	10^{-3}	4.387	2.048	2.203	0.942	1.030	0.171	2.028	3.565	1.877	2.996	17007.954	12138.284	62.762	152.343	0.012
		1	4.197	1.927	2.111	0.862	1.008	0.089	2.183	3.916	1.947	3.249	16415.939	12813.457	76.698	349.497	0.929
		10^3	3.160	1.314	1.565	0.496	1.000	0.000	2.417	4.744	1.889	3.265	12381.536	12176.535	55.780	349.204	4.936
1.0	10^{-3}	0.5	4.773	2.323	2.456	1.147	1.150	0.357	1.620	2.647	1.461	2.532	18184.687	12753.914	68.964	157.676	0.002
		1	4.150	1.881	2.089	0.840	1.000	0.000	2.140	4.147	1.844	3.164	15957.466	13294.614	79.743	219.785	1.299
		10^3	3.206	1.571	1.585	0.832	1.000	0.000	2.295	4.122	1.952	3.063	12468.135	12605.458	78.136	165.882	1.577
1.0	10^{-3}	1.0	2.035	0.857	1.001	0.011	1.000	0.000	3.195	5.697	2.258	3.775	11032.188	13271.551	95.347	293.532	1.753
		1	2.031	0.841	1.000	0.000	1.000	0.000	2.965	5.522	2.241	3.666	10662.381	13079.923	134.539	186.913	3.705
		10^3	2.052	0.879	1.003	0.053	1.000	0.000	3.151	5.561	2.413	4.200	11199.416	13729.502	94.664	198.483	4.194

Table F.3: Experimental results for the multi-specialty problem: patients' waiting time and overtime of ORs

γ	Variable	ENT	OBGYN		ORTHO		NEURO		GEN		OPHTH		VASCULAR		CARDIAC		UROLOGY	
			$u=1$	$u=3$	$u=1$	$u=3$	$u=1$	$u=3$	$u=1$	$u=2$	$u=1$	$u=2$	$u=1$	$u=4$	$u=1$	$u=2$	$u=1$	$u=2$
0.00	$\bar{\omega}_{ju}$	1.000	1.740	1.000	1.427	1.000	3.900	1.000	1.000	1.000	1.000	1.000	2.700	1.511	1.000	6.063	2.955	1.000
	$\sigma(\omega_{ju})$	0.000	0.417	0.000	0.280	0.000	0.709	0.000	0.000	0.000	0.000	0.000	1.750	0.342	0.000	0.371	0.043	0.000
	\bar{o}_j	0.000	0.052		0.015		1.097		0.086		0.000		0.000	0.052		0.369		0.000
	$\sigma(o_j)$	0.000	0.284		0.145		2.062		0.604		0.000		0.291			0.941		0.000
0.99	$\bar{\omega}_{ju}$	1.000	1.358	1.000	1.213	1.000	2.294	1.000	1.000	1.000	1.000	1.000	1.509	1.000	1.000	1.971	1.015	1.000
	$\sigma(\omega_{ju})$	0.000	0.436	0.000	0.167	0.000	1.270	0.000	0.000	0.000	0.000	0.000	0.341	0.000	0.000	0.885	0.015	0.000
	\bar{o}_j	0.000	0.004		0.216		1.304		0.044		0.000		0.000	0.740		0.743		0.000
	$\sigma(o_j)$	0.000	0.039		0.903		1.989		0.229		0.000		1.707			7.738		0.000

Table F.4: Experimental results for the multi-specialty problem: overuse of OR and SICU, cost and CPU time

γ	\bar{o}	$\sigma(o)$	\bar{e}	$\sigma(e)$	\bar{c}	$\sigma(c)$	\bar{t}	$\sigma(t)$
0.00	1.672	2.381	7.913	9.027	63714.601	21485.249	729,490	1078,424
0.99	3.050	3.271	8.690	9.518	21012.438	11487.441	663,610	736,206