RAIRO-Oper. Res. 53 (2019) 1819–1841
https://doi.org/10.1051/ro/2018110

**RAIRO Operations Research**
www.rairo-ro.org

# A MARKOV DECISION MODEL WITH DEAD ENDS FOR OPERATING ROOM PLANNING CONSIDERING DYNAMIC PATIENT PRIORITY

Jian Zhang[1,*], Mahjoub Dridi[1] and Abdellah El Moudni[1]

**Abstract.** This paper addresses an operating room planning problem with surgical demands from both the elective patients and the non-elective ones. A dynamic waiting list is established to prioritize and manage the patients according to their urgency levels and waiting times. In every decision period, sequential decisions are taken by selecting high-priority patients from the waiting list to be scheduled. With consideration of random arrivals of new patients and uncertain surgery durations, the studied problem is formulated as a novel Markov decision process model with dead ends. The objective is to optimize a combinatorial cost function involving patient waiting times and operating room over-utilizations. Considering that the conventional dynamic programming algorithms have difficulties in coping with large-scale problems, we apply several adapted real-time dynamic programming algorithms to solve the proposed model. In numerical experiments, we firstly apply different algorithms to solve the same instance and compare the computational efficiencies. Then, to evaluate the effects of dead ends on the policy and the computation, we conduct simulations for multiple instances with the same problem scale but different dead ends. Experimental results indicate that incorporating dead ends into the model helps to significantly shorten the patient waiting times and improve the computational efficiency.

**Mathematics Subject Classification.** 90B36, 90B50, 90C39, 90C40.

Received March 15, 2018. Accepted November 25, 2018.

## 1. Introduction

Hospitals are facing a continuously growing demand for health service due to the ageing population and the increasing quality of life [1]. This trend poses a complicated problem for the managers of medical institutions to meet the conflicting objectives of optimization (*e.g.*, reduce the patients' waiting time and optimize the utilization of facilities) while the medical resources are limited. In a hospital, operating rooms (ORs) consume the largest part of the budget and generate the highest revenue [2], as a result, OR planning has become a crucial activity to maximize the patient satisfaction and relieve the financial burden [3].

OR planning is a challenging task due to various uncertainties, different patient characteristics and conflicting interests of the major stakeholders (patients and hospitals). Uncertainties contribute to the high complexity of OR planning problems but cannot be neglected because they impose great effects on the balance between the

*Keywords.* Operating room planning, Markov decision process, dead end, real-time dynamic programming.

[1] Laboratoire Nanomédecine, Imagerie et Thérapeutiques, Université Bourgogne Franche-Comté, UTBM, Rue Thierry Mieg, 90010 Belfort cedex, France.
*Corresponding author: jian.zhang@utbm.fr

patient waiting times and the hospital expenses [4]. Various types of uncertainty are considered in literature, including duration uncertainty, arrival uncertainty, resource uncertainty and care requirement uncertainty [5]. Among these uncertainties, patient arrivals and surgery durations are two major sources of uncertainty frequently addressed [6–8]. These two stochastic aspects are taken into consideration in this paper to keep the model close to the reality. With respect to the categorization of patients, many researchers divide the patients into two groups: electives and non-electives (emergencies) [8, 9]. Elective cases can be postponed or scheduled ahead, whereas non-elective cases need to be performed as soon as possible. Instead of shunting the non-electives to dedicated ORs, we apply a flexible policy in this work that all the ORs are identical and versatile for both the electives and the non-electives, since [10] show that maintaining versatile ORs results in less delay of emergent cases. Besides, performance measures are the criteria to evaluate the quality of management and guide the OR managers to make decisions. Each performance measure favors the interest of one stakeholder over the other. This research focuses on the patient waiting times and the overtime of ORs, because long waiting time directly reduces the life quality of the patients and over-utilization of ORs results in dissatisfaction of the surgical staff and high expenses [4, 8, 11].

Researches on OR planning can be classified by three hierarchical decision levels: strategic level, tactical level and operational level [5,8,12]. This paper deals with a dynamic advanced scheduling problem for elective patients and emergency patients at the operational level. Elective surgeries are planned in advance while emergency surgeries must be served on the day of arrival due to their high urgency level. Decisions are made sequentially to determine the surgery date of each patient. Specifically, at the end of each decision period, we decide which patients in the waiting list will be served in the next period and how much OR capacity will be reserved for stochastic emergency demand. It is supposed that the regular OR capacity in each decision period has been fixed by a strategic plan. We do not classify the patients by specialties, and the scheduling of surgeons or surgical groups are not considered, so that a master surgical schedule that defines the allocation of ORs for multiple specialties or surgical groups is not needed. This work mainly optimizes the utilization of ORs and does not consider the planning of surgical staffs or upstream/downstream resources (*e.g.*, recovery beds and post-anaesthesia care units). Moreover, since the studied problem is a capacity planning problem, the assignments of patients to definite ORs are not implemented in this work, hence we concern the total capacity of ORs instead of the number of ORs. These problem settings are reasonable since they are close to existing researches such as [6, 13, 14]. Arrivals of patients (involving both the electives and the non-electives) and surgery durations are considered as stochastic variables in the model and generated via Monte-Carlo method in the simulations. A dynamic waiting list is established to schedule the elective patients. We pursue a balance between shorter waiting times of patients and lower over-utilization of ORs to maximize the satisfaction of both the patients and the hospital.

To solve the OR planning problems with stochastic factors, various mathematical methods are applied in literature, *e.g.*, mixed integer programming [15–17], linear programming [18, 19], column generation [20], constraint programming [21, 22], robust optimization [16, 23] and Markov decision process (MDP) [13, 24–27]. In this paper, we formulate the OR planning problem as a model of stochastic shortest path MDP with avoidable dead ends (SSPADE), an extension of the original MDP. To the best of our knowledge, no other studies have applied this formulation in the OR planning problems. Stochastic shortest path (SSP) MDP is a widely studied probabilistic planning problem with the objective of finding the optimal path to the goal state [28]. A major limitation of the SSP MDPs is its incapability of coping with many real problems with dead ends, states from which reaching the goal is impossible or too costly. Kolobov *et al.* [29] extend the concept of SSP MDP to SSPADE, to incorporate the dead ends into the decision problem. In our model, the set of dead ends contains the states which might lead to undesirable situations, *e.g.*, long waiting times of the patients and high expense of the hospital. When making decisions, the agent should avoid the dead ends to prevent the situation from being highly unwanted. The state with empty waiting list is selected as the goal. In practice, there are always new arrived patients even if the goal is reached (when all the patients in the waiting list are served), thus the horizon of the decision process is infinite. The SSPADE model is solved by the algorithms derived from real-time dynamic programming (RTDP) to improve the computational efficiency. By minimizing a cost function

combining the patient waiting cost and the overtime cost of ORs, we seek for the best trade-off between the interests of the patients and the hospital.

The contributions of this paper are summarized as follows. Firstly, a novel SSPADE formulation is adopted to model the OR planning problem. The undesirable states are regarded as dead ends and are avoided by the agent, so that the policy and the computational efficiency can be improved. Secondly, we assign each elective patient a time-dependent priority, allowing the ORs to be used overtime when there are too many high-priority patients in the waiting list. Elective patients directly benefit from this policy as their waiting times can be shortened. Finally, due to the fact that the RTDP-based algorithms are more efficient than the classical dynamic programming (DP) algorithms in solving the MDPs, we adapt the RTDP and its variants to solve the SSPADE model.

The reminder of this paper is organized as follows. Section 2 presents a literature review on the problem we are studying. Section 3 proposes the SSPADE model for the OR planning problem. Section 4 introduces the adapted RTDP-based algorithms. Results of the numerical experiments are presented in Section 5. We compare the computational efficiencies of the applied algorithms and analyze the extent to which different dead end settings influence the policy and the computational efficiency. Section 6 provides the final conclusion of this research and suggests the future works.

## 2. Literature review

OR planning has drawn considerable interest of researchers and many articles on this topic can be found. The latest literature reviews are presented by Cardoen *et al.* [11], Guerriero and Guido [12], Van Riet and Demeulemeester [4], Samudra *et al.* [8] and Zhu *et al.* [5]. In literature, most researchers focus on the scheduling of elective patients, whilst only limited researches are done by incorporating non-elective patients due to the larger degree of uncertainty [4,11]. To find a balance between shortening waiting times of electives and ensuring quick-access for emergencies, three allocation policies have been proposed in existing works: in the dedicated policy, a subset of ORs are dedicated exclusively to performing emergency surgeries while the others are used to serve elective patients [30,31]; in the flexible policy, all the ORs are identical and no OR is specially reserved for non-electives [32, 33]; the hybrid policy is a trade-off of the two policies mentioned above and maintains both versatile ORs and dedicated ORs [26, 34]. Ferrand *et al.* [10] compare the patient waiting times and the OR over-utilization by evaluating both a dedicated and a flexible policy via simulation, and they conclude that the former results in lower elective waiting times and less overtime of the ORs, but significantly increases the emergency waiting times. In this work, we adopt a flexible policy that all the ORs are accessible to both the electives and the non-electives, to guarantee responsiveness and quick-access for the most urgent cases.

Despite of higher randomness, planning non-elective surgeries is relatively simple and a first-come-first-serve (FCFS) way is often applied [10,33], whereas when scheduling elective patients, it is necessary to prioritize them according to their different urgency levels. Among the researches about OR planning and patient scheduling reviewed by Van Riet and Demeulemeester [4], most authors do not mention their prioritization system. One of the exceptions is that Patrick *et al.* [24] deal with a dynamic patient scheduling problem with multiple patient priorities. A cost increasing in priority level and waiting time is incurred if a patient is not served before the maximum recommended waiting time is reached. Moreover, Testi *et al.* [35] introduces a prioritization scoring system to define an admission rule. The product of urgency level and waiting time is regarded as the score to quantify the patient priority. Valente *et al.* [36] propose a pre-admission process model to prioritize elective patients. An assessment of clinical urgency is carried out when a patient comes into the waiting list, then an MTBT (maximum time before treatment) and a real-time priority in accordance with urgency level and waiting time are assigned to the patient. The prioritization methods in these two works have been put into practice in Italian hospitals and the results confirm their efficiency and equity. Min and Yih [6] addresses an elective patient scheduling problem with patient priorities considered. The optimal policy is derived from the trade-off between the cost of overtime work and the cost of surgery postponement. They evaluate different ratios of the patient waiting cost to the overtime cost, and come to the conclusion that this ratio should not be low for fear that the scheduling might be marginally affected by prioritizing. Min and Yih [13] adopt the prioritization scoring system

introduced by Testi *et al.* [35] to manage an elective patient waiting list, and go one step further by incorporating dynamic patient health status. Their model is capable of minimizing a combined cost function which captures overtime work, waiting time as well as patient related adverse events such as deterioration of health and death. The ORs are allowed to be used longer than regular work time when there are many high-priority patients in the waiting list, so that the surgery postponement costs are balanced with the overtime costs.

Uncertainty is an intrinsic property of the OR planning problems. Four categories of uncertainty are considered by researchers [5]: duration uncertainties mainly include uncertain surgery duration and uncertain length of stay (LOS) [7,9,16,23,37]; Arrival uncertainties involve the random arrivals of elective and non-elective patients [6,13,14,27]; resource uncertainties refer to the uncertain availability of human resources or material resources [15,36–38]; care requirement uncertainties refer to the uncertain patient demands for surgical care [39,40]. The literature on stochastic OR planning exhibits a variety of operations research methodologies. Wang *et al.* [20] model an OR planning problem with uncertain surgery duration and emergency demand as an integer programming problem. They introduce a column-generation-based heuristic algorithm to solve the problem efficiently. Saadouli *et al.* [15] study a scheduling problem for elective patients with uncertainties in surgery, recovery durations and capacity of resources. A two-phase solution procedure combining mixed integer programming and discrete event simulation is proposed in their work. Latorre-Núñez *et al.* [18] solve an OR scheduling problem by developing a tool including mixed integer linear programming (MILP), constraint programming and meta-heuristics. Computational experiments show that both of these methods deliver high-quality solutions and the meta-heuristic approach consumes the shortest CPU time. Heydari and Soudi [19] address a predictive/reactive surgical suite scheduling problem for elective patients and emergencies. Uncertainties lie in the arrivals and the surgery times of emergent patients. A two-stage stochastic programming model with recourse is proposed to minimize the effect of disruption imposed on the system and preserve the stability and the robustness of the schedule. Marques and Captivo [16] manage a scheduling problem for elective patients with uncertain surgery duration. They model the problem as an MILP to optimize the use of surgical resources and improve equity and access to the patients. Burdett and Kozan [17] use a flexible job-shop scheduling (FJSS) formulation to plan hospital activities in an integrated way that ORs, beds and other treatment spaces except for the emergency department are taken into account. They develop constructive algorithms and hybrid meta-heuristics to solve the cases with real world size.

Apart from the methodologies mentioned above, Markov decision process (MDP) has a good performance in solving sequential decision-making problems with uncertainties, but the applications of MDP model in OR planning are limited. In the work of Patrick *et al.* [24], a dynamic patient scheduling problem with stochastic arrivals of patients is formulated as an MDP. Zonderland *et al.* [25] develop a decision supporting tool based on an infinite-horizon MDP to assist the scheduling of elective and semi-urgent surgeries. They conclude that the application of MDP substantially simplifies the scheduling task. Hosseini and Taaffe [26] use Markov decision model to find an optimal solution for arranging elective and non-elective surgeries under a hybrid policy. Min and Yih [13] address a problem of managing elective patient waiting list with time-dependent priority and adverse events. This problem is formulated as an infinite-horizon MDP to optimize the balance between patient waiting times and overtime of ORs. Astaraky and Patrick [27] establish an MDP model for a patient scheduling problem in a multi-class and multi-resource surgical system. They use real data from a hospital to evaluate the model and simulation results demonstrate the success of their policy.

As for solution techniques, dynamic programming (DP) algorithms such as value iteration (VI) and policy iteration (PI) are the fundamental methods to solve MDPs. A practical drawback of these algorithms is that they need to explore the entire state space repeatedly until the convergence is detected. The size of state space is often intractably large for realistic problems. Consequently, the memory size and the CPU time required for computation are often unacceptable. To improve the computational efficiency, Barto *et al.* [41] propose real-time dynamic programming (RTDP). This is a heuristic-search algorithm that provides a partial policy rather than a complete one. It solves MDPs by restricting the computations to the reachable states from the initial state. RTDP does not carry out full sweeps for the whole state space and saves both memory and time. Given that the original RTDP has no proper criterion to detect the convergence or terminate the computation,

Bonet and Geffner [42] introduce labelled RTDP (LRTDP) by adding a labelling scheme as the termination condition on the basis of the original RTDP. Both RTDP and LRTDP are guaranteed to converge if the value functions are initialized as lower bounds of the optimal value functions, but LRTDP converges much faster. Besides, McMahan *et al.* [43] propose bounded RTDP (BRTDP) which maintains an upper bound in addition to the lower bound. BRTDP uses the gap between the two bounds to detect the convergence and to guide the exploration in the state space, so that the process of convergence can be more uniform and faster. Similarly, Smith and Simmons [44] introduce focused RTDP (FRTDP), which is another extension of RTDP also maintaining two bounds. Furthermore, Sanner *et al.* [45] provide value of perfect information RTDP (VPI–RTDP). This approach improves on BRTDP and FRTDP by biasing the agent to evaluate the states where the policies are not converged. The main advantage of VPI–RTDP is that it does not waste computation on the states where policies are converged but value functions are not.

In this paper, we adopt the prioritization scoring system proposed and used by Valente *et al.* [36], Testi *et al.* [35] and Min and Yih [13], but our work differs from them by incorporating non-elective patients into the OR planning model. Flexible policy is used to handle the allocation of OR capacity for different groups of patients. A previous work [46] has addressed the same OR planning problem with a model of finite-penalty SSP MDP with dead ends (fSSPDE), which is solved by DP-based algorithms. Compared to [46], this work adopts a novel SSPADE formulation, which is more compactly and more explicitly defined than fSSPDE, to model the studied problem. More importantly, RTDP-based algorithms are employed in this work to overcome the weakness of DP in coping with large state space. These algorithms are adapted to fit the SSPADE model and their computational efficiencies are evaluated through numerical experiments.

## 3. Modeling of operating room planning

### 3.1. Problem configuration

In this work we consider an OR planning problem in a department where elective patients and non-elective ones share the same medical facilities. Non-elective patients are urgent cases that must be served on the day of arrival. Elective surgeries are planned in advance and we set up a waiting list to schedule them. At the end of each decision period, surgical service group selects the elective patients that will be served during the next period and determine the OR capacity reserved for emergencies. Table 1 summarizes the notations in the model.

A time-dependent prioritization system is applied to classify the elective patients. We assign a priority score $P_r$ integrating urgency level $u$ and waiting time $t$ to each patient.

$$P_r = u \times t \tag{3.1}$$

where $u \in \{1, 2, \ldots, U\}$ and $t \in \{1, 2, \ldots, T_u\}$. $U$ is the highest urgency level and $T_u$ is the maximum allowed waiting time of the patients at urgency level $u$. Every patient must be scheduled before the waiting time $t$ reaches $T_u$. Urgency level $u$ is determined by the medical service staff when a patient is added into the waiting list and this parameter does not change, while waiting time $t$ continuously increases until the patient is served. Subscript $\tau \in \mathbb{N}^+$ is defined as the period number. Then we use $N_u^\tau(t) \in \mathbb{N}^0$ to denote the number of patients at urgency level $u$ who came into the waiting list $t$ periods ago (*i.e.*, arrived in period $\tau - t$). Similarly, we use $m_u^\tau(t) \in \mathbb{N}^0$ to denote the number of patients with urgency level $u$ and waiting time $t$ that are scheduled to be served during period $\tau$.

Arrivals of new patients and surgery durations are subject to uncertainties. Poisson distribution is a classical assumption for patient arrivals in literature [4, 14, 27, 32, 34], and lognormal distribution is widely used to describe the stochasticity of surgery durations and results in the best fit for the hospital data [4, 16, 37, 47, 48]. Therefore, we assume that arrivals of new patients and surgery durations are Poisson and lognormal distributed, respectively. Let $n_u^\tau \in \mathbb{N}^0$ be the number of new arrived elective patients at urgency level $u$ in period $\tau$, and let $n_0^\tau \in \mathbb{N}^0$ be the number of arrived non-elective patients in period $\tau$. Then the transition of the waiting list from

TABLE 1. Summary of notations.

| Notation | Definition |
|---|---|
| $P_r$ | priority score of patient |
| $u$ | urgency level of patient |
| $t$ | waiting time of patient |
| $\tau$ | index of decision period |
| $U$ | highest patient urgency level |
| $T_u$ | maximum allowed waiting time of patients at urgency level $u$ |
| $N_u^\tau(t)$, $N_u(t)$ | number of patients at urgency level $u$ that have been waiting for $t$ periods |
| $m_u^\tau(t)$, $m_u(t)$ | number of scheduled patients at urgency level $u$ that have been waiting for $t$ periods |
| $n_u^\tau$, $n_0^\tau$ | number of new arrived electives at urgency level $u$ and arrived non-electives in $\tau$ |
| $c_w$, $c_o$ | unit waiting cost of patients and over-utilization cost of ORs |
| $C_w^\tau$, $C_o^\tau$, $C^\tau$ | waiting cost, overtime cost and total cost of period $\tau$ |
| $S$ | set of states |
| $s^\tau$, $s$ | state |
| $s_0$ | initial state |
| $A$ | set of actions |
| $a^\tau$, $a$ | action |
| $p(s, a, s')$ | transition probability from $s$ to $s'$ when $a$ is executed |
| $C(s, a)$ | cost function when $a$ is executed at state $s$ |
| $G$ | set of goal states |
| $D$ | set of dead ends |
| $S'$ | set of non-goal and non-dead-end states |
| $\pi$, $\pi^*$ | complete policy, optimal complete policy |
| $\pi_p$, $\pi_p^*$ | partial policy, optimal partial policy |
| $n$ | index of iteration (in VI) or index of trial (in RTDPs) |
| $\epsilon$ | threshold to terminate the computation ($\epsilon > 0$) |
| $V^*(s)$ | optimal value function of state $s$ |
| $V_n(s)$ | value function of state $s$ in literation $n$ |
| $V_u(s)$, $V_l(s)$ | upper bound and lower bound of state $s$ |
| $L_u$ | maximum number of patients at urgency level $u$ |
| $l_u(t)$ | maximum number of patients at urgency level $u$ that have been waiting for $t$ periods |

period $\tau$ to period $\tau + 1$ can be written as

$$\begin{cases} N_u^{\tau+1}(1) = n_u^\tau \\ N_u^{\tau+1}(t+1) = N_u^\tau(t) - m_u^\tau(t) \end{cases}. \tag{3.2}$$

Objectives are to reduce the waiting times of patients and to control the over-utilization of ORs, hence we define two types of costs as our performance measures: if a patient is still not served at the end of a decision period, a waiting cost of $c_w \times P_r$ is incurred ($c_w$ is the unit waiting cost); if the regular work time of the ORs is exceeded, an over-utilization cost of $c_o$ per hour is incurred.

### 3.2. Stochastic shortest path MDP with avoidable dead-ends

Conventional MDP formulation is not suitable for the problem described in Section 3.1, as the number of patients in the waiting list can grow to infinity and the state space is infinite. In this work, the studied problem is formulated as a solvable SSPADE model. The definition and notations regarding the SSPADE model are given in Definition 3.1 [49].

**Definition 3.1.** A stochastic shortest path MDP with avoidable dead ends (SSPADE) is a tuple $\langle S, A, p, C, G, D, s_0 \rangle$

- $S$ is the state space, *i.e.* the set of states $s$;
- $A$ is the action space, *i.e.* the set of actions $a$;
- $p \to S \times A \times S \to [0,1]$ is the stationary transition function specifying the probability $p(s,a,s')$ of transferring from state $s$ to state $s'$ if action a is executed;
- $C \to S \times A \to [0,+\infty)$ is the stationary cost function specifying the cost $C(s,a)$ of executing action $a$ at state $s$;
- $G \subseteq S$ is the set of goal states s.t. $\forall s \in G$ and $\forall a \in A$, the cost function obeys $C(s,a) = 0$;
- $D \subseteq S$ is the set of dead ends s.t. $\forall s \in D$, the probability of transferring to goal states within a finite number of steps is zero;
- $s_0$ is the initial state.

As defined above, a dead end is a state from which no policy can reach the goal state in a finite series of steps. In the previous work [46], a finite penalty $P$ ($P \in \mathbb{R}^+$ is a large number) is defined as the cost of visiting a dead end. While in the studied problem of this work, the dead ends should always be avoided by the agent. Therefore, to make the model description more compact, we assign an infinite cost to each dead end in this work. Then the Bellman equation of the SSPADE model is written as

$$V^*(s) = \begin{cases} 0 & \text{if } s \in G \\ \infty & \text{if } s \in D \\ \min\limits_{a \in A} \sum\limits_{s' \in S} p(s,a,s')[C(s,a) + V^*(s')] & \text{otherwise} \end{cases}. \tag{3.3}$$

A complete policy $\pi : S \to A$ is a mapping that specifies the corresponding action $a = \pi(s)$ for every state $s$. Solving an SSPADE model means finding the optimal policy $\pi^*$ that can avoid all the dead ends as well as reach the goal state with the lowest cost.

$$\pi^*(s) = \arg\min_{a \in A} V^*(s). \tag{3.4}$$

In the SSPADE model for OR planning, state $s$ and action $a$ are defined as sets of $N_u(t)$ and $m_u(t)$.

$$s = \{N_u(t) \in \mathbb{N}^0 | u, t \in \mathbb{N}^+, u \leqslant U, t \leqslant T_u\} \tag{3.5}$$

$$a = \{m_u(t) \in \mathbb{N}^0 | u, t \in \mathbb{N}^+, u \leqslant U, t \leqslant T_u\}. \tag{3.6}$$

Superscript $\tau$ can be removed here to keep the notations simple because SSPADE is a stationary model (transition function, value function or policy does not depend on period number $\tau$). Then every action $a$ that satisfies the following condition is applicable to a given state $s$: $\forall u \in [1, U]$,

$$\begin{cases} m_u(t) \leqslant N_u(t), \text{ if } 1 \leqslant t \leqslant T_u - 1 \\ m_u(t) = N_u(t), \text{ if } t = T_u \end{cases}. \tag{3.7}$$

Transition probability from period $\tau$ to $\tau + 1$ can be calculated by

$$\begin{aligned} p(s^\tau, a^\tau, s^{\tau+1}) = &\{\prod_{u=1}^{U} p[N_u^{\tau+1}(1) = n_u^\tau]\} \\ &\times \{\prod_{u=1}^{U} \prod_{t=2}^{T_u} p[N_u^{\tau+1}(t) = N_u^\tau(t-1) - m_u^\tau(t-1)]\}. \end{aligned} \tag{3.8}$$

Waiting cost and over-utilization cost of period $\tau$ are given as follows:

$$C_w^\tau = \sum_{u=1}^{U} \sum_{t=1}^{T_u} c_w P_r[N_u^\tau(t) - m_u^\tau(t)] \tag{3.9}$$

$$C_o^\tau = c_o \times \max\{0, H^\tau - H\}. \tag{3.10}$$

where $H^\tau$ is the total surgery time of period $\tau$ and $H$ is the regular open time of the ORs in the department. Then the total cost of period $\tau$ is $C^\tau = C_w^\tau + C_o^\tau$.

According to Definition 3.1, goal state set $G$ and dead end set $D$ are two subsets of the state space $S$. We define the state with empty waiting list as the only element of $G$, *i.e.*, for the goal state $s \in G$, $\forall u \in [1, U]$ and $\forall t \in [1, T_u]$, $N_u(t) = 0$. This means that the goal of the proposed model is to serve all the patients with the lowest cost. No action needs to be taken in the goal state. If the decision process is terminated by reaching the goal state, new patients will continue to arrive, then a new round of decision process will begin. Therefore, the decision horizon of this OR planning problem is infinite.

Unwanted states are classified into the set of dead ends. For a state $s$, the conditions of being a non-dead-end state are given as follows: $\forall u \in [1, U]$ and $\forall t \in [1, T_u]$

$$\sum_{t=1}^{T_u} N_u(t) \leqslant L_u \in \mathbb{N}^+ \tag{3.11}$$

$$N_u(t) \leqslant l_u(t) \in \mathbb{N}^+ \tag{3.12}$$

where $L_u$ gives an upper limit of the total number of patients at level $u$, and $l_u(t)$ restricts the number of patients at level $u$ with waiting time $t$. If all the inequalities given by (3.11) and (3.12) hold, $s$ is a non-dead-end state; on the contrary, if one or several of these conditions are not satisfied, $s$ is a dead end. Theoretically, the agent might still be able to reach the goal state after visiting such a dead end, but we artificially assume that this probability is zero, to force the agent to avoid these unwanted states. For non-dead-end states, the numbers of elective patients in the waiting list are limited. To prevent from encountering a dead end, the agent needs to reduce the patients in the waiting list by keeping the ORs open longer than the regular work time if necessary. As a result, incorporating dead ends into the model might contribute to the reduction of elective waiting times.

To the best of our knowledge, the application of SSPADE model in OR planning has not been researched in literature. We propose two principles that should be followed when determining the dead end parameters $l_u(t)$ and $L_u$.

$$l_u(t+1) \leqslant l_u(t), \forall u \in [1, U], \forall t \in [1, T_u] \tag{3.13}$$

$$l_u(1) \leqslant L_u \leqslant \sum_{t=1}^{T_u} l_u(t), \forall u \in [1, U]. \tag{3.14}$$

It is assumed that $l_u(T_u + 1) = 0$. Obviously, a patient becomes more undesirable to stay in the waiting list as the waiting time increases, hence inequality (3.13) defines that $l_u(t)$ decreases monotonically in $t$. Besides, $L_u$ should not be larger than the sum of $l_u(t)$ to ensure the effectiveness of $L_u$, as defined by the left part of (3.14). Similarly, the right part of (3.14) guarantees the effectiveness of $l_u(t)$. Following these principals, the dead end parameters can be roughly determined in accordance with the department scale and the arriving rates of patients, or based on the experience of the medical staff. In Section 5, we compare the experimental results obtained from a series of simulations with different dead end parameters to analyze the effects of dead ends.

### 3.3. Model analysis and adapted value iteration

For an SSP MDP, there exists at least one complete proper policy $\pi$ that allows the agent to reach the goal from any other state in a finite number of steps with a finite cost. While for an SSPADE, no policy can take the agent from a dead end to the goal state. Although the dead ends can be avoided, they still exist in the state space and their value functions are $\infty$. Now that VI operating on the whole state space does not converge, to make the SSPADE model feasible for our problem, we restrict the computation within the subset $S' = S \backslash (D \cup G)$ which excludes the dead ends and the goal state. We are only interested in finding an optimal partial policy $\pi_p^*$ for the domain of $S'$. The existence of a partial policy $\pi_p$ is given by following proposition.

**Proposition 3.2.** *There exists at least one partial policy $\pi_p$ that is closed to every $s \in S'$, i.e., $\forall s \in S'$, any state $s'$ reachable with policy $\pi_p$ from $s$ is contained in subset $S'$, only if $\forall u \in [1, U]$ and $\forall \tau \in \mathbb{N}^+$, $n_u^\tau \leqslant l_u(1)$ holds.*

*Proof.* Consider that the agent moves from state $s^\tau = \{N_u^\tau(t)\}$ to state $s^{\tau+1} = \{N_u^{\tau+1}(t)\}$ by executing action $a^\tau = \{m_u^\tau(t)\}$. According to the dead end conditions defined by (3.11) and (3.12), if the new arrived patients $n_u^\tau > l_u(1)$, then $N_u^{\tau+1}(1) > l_u(1)$, $s' \notin S'$. Therefore, $n_u^\tau \leqslant l_u(1)$ is a necessary condition. Now given that $N_u^{\tau+1}(1) = n_u^\tau \leqslant l_u(1)$ holds, if $s^{\tau+1} \in S'$, then following inequalities hold: $\forall u \in [1, U]$ and $\forall t \in [1, T_u]$,

$$\begin{cases} N_u^{\tau+1}(t+1) = N_u^\tau(t) - m_u^\tau(t) \leqslant l_u(t+1) \\ \sum_{t=1}^{T_u} N_u^{\tau+1}(t) = \sum_{t=1}^{T_u} N_u^\tau(t) - \sum_{t=1}^{T_u} m_u^\tau(t) + n_u^\tau \leqslant L_u \end{cases}. \tag{3.15}$$

The stochastic variable $n_u^\tau$ can be removed by considering $n_u^\tau \leqslant l_u(1)$, then the action $a^\tau$ only need to meet following requirements to keep $s' \in S'$ true:

$$\begin{cases} m_u^\tau(t) \geqslant N_u^\tau(t) - l_u(t+1) \\ \sum_{t=1}^{T_u} m_u^\tau(t) \geqslant \sum_{t=1}^{T_u} N_u^\tau(t) + l_u(1) - L_u \end{cases}. \tag{3.16}$$

Evidently, these requirements do not contradict the conditions defined by (3.7). That is, for any non-dead-end state $s$, there exists at least one proper action $a$ that prevents the agent from visiting a dead end. Therefore, a set of proper actions can form a proper partial policy $\pi_p$ that is closed in the domain of $S'$. □

Combining (3.7) and (3.16), then all the requirements for action $a$ can be written as follows: $\forall u \in [1, U]$,

$$\begin{cases} N_u^\tau(t) - l_u(t+1) \leqslant m_u^\tau(t) \leqslant N_u^\tau(t), & \text{if } 1 \leqslant t \leqslant T_u - 1 \\ m_u(t) = N_u(t), & \text{if } t = T_u \\ \sum_{t=1}^{T_u} m_u^\tau(t) \geqslant \sum_{t=1}^{T_u} N_u^\tau(t) + l_u(1) - L_u \end{cases}. \tag{3.17}$$

For each state-action pair, if there is one or several dead ends among its successors, this pair will be abandoned by the agent due to the infinite cost. In this way, the agent can avoid the dead ends and reach the goal state with probability 1.

VI is a widely used DP method for solving MDP problems and it is the basis of many advanced algorithms. As Proposition 3.2 confirms the existence of $\pi_p$, an adapted version of VI can be applied to the proposed SSPADE model. The procedure is shown in Algorithm 1.

To ameliorate the computational efficiency, we execute the Bellman equation (line 8) in an asynchronous way: all the calculated values of the current iteration are available immediately to the following computations, *i.e.*, the successor values could be from the current iteration itself. With this algorithm, we can get an optimal partial policy $\pi_p^*$ by focusing the computation on the subset $S'$.

## 4. Real-time dynamic programming approaches

The major drawback of VI is that it needs to fully sweep the state space and the action space. The amount of computation and the demand for memory size increase exponentially as the problem scale becomes larger, rendering VI impractical for many real problems. RTDP is an on-line heuristic-search method for MDP models that does not evaluate the entire state space. It can often deliver an optimal partial policy $\pi_p^*$ with respect to the initial state $s_0$ and consume less CPU time than VI. In this work, we consider several RTDP-based approaches and investigate their performances.

**Algorithm 1:** Adapted Value Iteration for SSPADE model.

$\forall n \in \mathbb{N}^0$, $s \in D$, $V_n(s) = \infty$;
$\forall n \in \mathbb{N}^0$, $s \in G$, $V_n(s) = 0$;
$\forall s \in S'$, $V_0(s) = 0$;
iteration $n = 0$;
**while** $(n = 0) \vee (|V_n(s) - V_{n-1}(s)| \geqslant \epsilon)$ **do**
  $n \leftarrow n + 1$;
  **forall the** $s \in S'$ *and* $a \in A$ **do**
    $V_n(s) = \min_{a \in A} \sum_{s' \in S} p(s, a, s')[C(s, a) + V_n(s')]$;
    $\pi_p^*(s) = \arg\min_{a \in A} V_n(s)$;
obtain the policy $\pi_p^* = \{\pi_p^*(s) | s \in S'\}$;

### 4.1. RTDP approaches with lower bound

RTDP and its variants are all based on a mechanism of FIND-REVISE: FIND means that they continuously simulate the current optimal policy $\pi_p^*$ to sample the next state $s'$ which is to be visited; REVISE means that they execute the Bellman equation at every visited state to renew the value function and the policy. The adapted RTDP algorithm for the SSPADE model is presented in Algorithm 2.

**Algorithm 2:** Adapted RTDP for SSPADE Model.

$\forall s \in D$, $V_l(s) = \infty$;
$\forall s \in G \bigcup S'$, $V_l(s) = 0$;
$n = 0$;
**while** $(n \leqslant N) \wedge (s_0 \notin G)$ **do**
  $s \leftarrow s_0$;
  $n \leftarrow n + 1$;
  $depth = 0$;
  *//trial*
  **while** $(depth \leqslant max\_depth) \wedge (s \notin G)$ **do**
    $depth = depth + 1$;
    $V_l(s) = \min_{a \in A} \sum_{s' \in S} p(s, a, s')[C(s, a) + V_l(s')]$;
    $a^* = \pi_p^*(s) = \arg\min_{a \in A} V_l(s)$;
    choose $s'$ randomly by executing $a^*$ in $s$;
    $s \leftarrow s'$;
return $\pi_p^*(s_0)$;

It has been proved that the algorithms based on the FIND-REVISE mechanism are capable of solving SSPADE models [29]. The convergence of the original RTDP is guaranteed by initializing the value function as a monotone lower bound $V_l(s)$: for any $s \in S$, $V_l(s) \leqslant \min_{a \in A} \sum_{s' \in S} p(s, a, s')[C(s, a) + V_l(s')]$, *i.e.*, $V_l(s) \leqslant V^*(s)$ and $V_l(s)$ can only increase when a Bellman equation is executed. When applying RTDP to solve our model, the computation is still restricted in the subset $S'$.

Though RTDP is guaranteed to converge at the optimal value function $V^*(s)$, it lacks a detection of convergence or a proper condition to terminate the computation. This weakness can be avoided by assigning each state a label which indicates whether the value function of the state is converged or not. Bonet and Geffner [42] introduce LRTDP approach which incorporates a labelling scheme into the original RTDP. LRTDP also

maintains a lower bound and works mostly the same as RTDP does. The labelling scheme marks a state $s$ as *solved* if the value functions of $s$ and all its descendants are $\epsilon$-*consistent*. The value function of a state $s$ is $\epsilon$-*consistent* if its residual between two consecutive iterations is less than $\epsilon$, *i.e.* $Res^{V_l}(s) < \epsilon$. The adapted version of LRTDP for the SSPADE model is presented in Algorithm 3. The mechanism of convergence detection is realized in sub-function $CHECKSOLVED$, which is not given in this paper but can be found in [42].

**Algorithm 3:** Adapted LRTDP for SSPADE Model.

$\forall s \in D$, $V_l(s) = \infty$;
$\forall s \in G$, $V_l(s) = 0$, labeled as solved;
$\forall s \in S'$, $V_l(s) = 0$, labeled as not solved;
**while** $s_0$ *is not solved* **do**
  $s \leftarrow s_0$;
  $visited \leftarrow EMPTYSTACK$;
  *//trial*
  **while** $s$ *is not solved* **do**
    push $s$ onto *visited*;
    $V_l(s) = \min_{a \in A} \sum_{s' \in S} p(s, a, s')[C(s, a) + V_l(s')]$;
    $a^* = \pi_p^*(s) = \arg\min_{a \in A} V_l(s)$;
    choose $s'$ randomly by executing $a^*$ in $s$;
    $s \leftarrow s'$;
  **while** $visited \neq EMPTYSTACK$ **do**
    pop $s$ from *visited*;
    **if** $CHECKSOLVED(s, \epsilon) \neq true$ **then** break;
return $\pi_p^*(s_0)$;

## 4.2. RTDP approaches with lower and upper bounds

Despite the fact that LRTDP improves the speed of convergence by applying the labelling mechanism, it still has a drawback: when choosing the next state $s'$ to update the value function and the policy, LRTDP randomly samples it from the successors of the state-action pair $\{s, a^*\}$ according to the transition function $p(s, a^*, s')$. The convergence can be faster and more uniform if those less converged states are more likely to be selected and visited. McMahan *et al.* [43] improve RTDP and LRTDP by adding an upper bound $V_u(s)$ in addition to the lower bound $V_l(s)$ and introduce BRTDP approach. Similar to the definition of lower bound, a monotone upper bound satisfies $V_u(s) \geqslant \min_{a \in A} \sum_{s' \in S} p(s, a, s')[C(s, a) + V_u(s')]$ for any $s \in S$, *i.e.*, $V_u(s) \geqslant V^*(s)$ and the value function initialized as an upper bound can only decrease when a Bellman equation is executed. Considering that $V^*(s) \in [V_l(s), V_u(s)]$, the gap between the two bounds $V_u(s) - V_l(s)$ indicates the extent of convergence: the smaller this gap is, the better the state $s$ is understood. Then $V_u(s) - V_l(s) < \epsilon$ can be used as the criterion to terminate the computation. When choosing the next state $s'$ for a state $s$ with the current best action $a^*$, BRTDP samples it with the probability proportionate to $p(s, a^*, s')[V_u(s') - V_l(s')]$. Therefore, maintaining two bounds on the optimal value function provides a good guarantee of performance, and allows us to focus on those states which are both relevant and poorly understood.

Another extension of RTDP which also maintains two bounds is FRTDP proposed by Smith and Simmons [44]. For our problem, both BRTDP and FRTDP are adapted to fit the SSPADE model and are shown in Algorithms 4 and 5, respectively. It can be seen from Algorithm 5 that, instead of sampling the next state $s'$ from a distribution, FRTDP uses the product of the transition possibility $p(s, a^*, s')$ and the priority function $prio(s')$ to deterministically choose the state with the highest value. Moreover, FRTDP measures the update

**Algorithm 4:** Adapted BRTDP for SSPADE Model.

$\forall s \in D$, $V_l(s) = \infty$, $V_u(s) = \infty$;
$\forall s \in G$, $V_l(s) = 0$, $V_u(s) = 0$;
$\forall s \in S'$, $V_l(s) = 0$, $V_u(s) = V_u^0$; *//$V_u^0$ is the initial upper bound*
**while** $V_u(s_0) - V_l(s_0) \geqslant \epsilon$ **do**
    $s \leftarrow s_0$;
    *//trial*
    **while** $V_u(s) - V_l(s) \geqslant \epsilon$ **do**
        $V_u(s) = \min_{a \in A} \sum_{s' \in S} p(s, a, s')[C(s, a) + V_u(s')]$;
        $V_l(s) = \min_{a \in A} \sum_{s' \in S} p(s, a, s')[C(s, a) + V_l(s')]$;
        $a^* = \pi_p^*(s) = \arg\min_{a \in A} V_l(s)$;
        $\forall s' \in S'$, $b(s') = p(s, a^*, s')[V_u(s') - V_l(s')]$;
        **if** $\sum_{s'} b(s') < [V_u(s_0) - V_l(s_0)]/\eta$ **then** break;
        **else**
            choose $s'$ from distribution $b(s')/\sum_{s'} b(s')$;
            $s \leftarrow s'$;
return $\pi_p^*(s_0)$;

quality $q$ of each REVISE and adjust the depth of the *trial* loop: at the end of each *trial* loop (when $d > D/k_D$), if the average update quality ($q_{\text{curr}}/n_{\text{curr}}$) is not worse than that in the earlier part of the *trial* ($q_{\text{prev}}/n_{\text{prev}}$), the maximum depth of the *trial* loop will be updated by $D = k_D D$.

### 4.3. VPI–RTDP

Maintaining two bounds allows both $V_u(s)$ and $V_l(s)$ to converge to the optimal value function $V^*(s)$ uniformly and quickly, but BRTDP and FRTDP might waste time on evaluating the states whose policies are already converged while value functions are not. Sanner *et al.* [45] improve the RTDP algorithm by proposing a value of perfect information (VPI) analysis to guide the exploration in the state space, and their algorithm is named VPI–RTDP. VPI–RTDP maintains an upper bound and a lower bound on the optimal value function and carries out a VPI analysis to estimate the expected improvement of policy by visiting a state. The exploration in the state space is directed to the states whose value updates may result in greater policy improvement, *i.e.*, the states with higher VPI values. The detailed procedure of VPI analysis is given in this section.

For $s \in G$ and $s \in D$, their value functions are constantly 0 and $\infty$, respectively. Considering that $V^*(s)$ for $s \in S'$ is unknown before the bounds are converged, and $V^*(s) \in [V_l(s), V_u(s)]$, it can be assumed that $V^*(s)$ is uniformly distributed in the interval $[V_l(s), V_u(s)]$, then the probability density function of $V^*(s)$ for $s \in S'$ is

$$f[V^*(s)|\overrightarrow{V}] = \begin{cases} \dfrac{1}{V_u(s) - V_l(s)}, & V^*(s) \in [V_l(s), V_u(s)] \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

where $\overrightarrow{V} = \{\overrightarrow{V_u}, \overrightarrow{V_l}\}$.

The expected Q-value of state-action pair $\{s, a\}$ is

$$\begin{aligned} \mathbb{E}[Q(s, a)|\overrightarrow{V}] &= \mathbb{E}\{C(s, a) + \sum_{s'' \in S} p(s, a, s'')[V^*(s'')]\} \\ &= C(s, a) + \sum_{s'' \in S} p(s, a, s'') \frac{V_l(s'') + V_u(s'')}{2}. \end{aligned} \tag{4.2}$$

**Algorithm 5:** Adapted FRTDP for SSPADE Model.

$\forall s \in D$, $V_l(s) = \infty$, $V_u(s) = \infty$;
$\forall s \in G$, $V_l(s) = 0$, $V_u(s) = 0$;
$\forall s \in S'$, $V_l(s) = 0$, $V_u(s) = V_u^0$, $prio(s) = \Delta(s) = V_u(s) - V_l(s) - \epsilon/2$;
$D = D_0$;//*maximum depth of trial is initialized as* $D_0$
**while** $V_u(s_0) - V_l(s_0) \geqslant \epsilon$ **do**
  $[q_{\text{prev}}, n_{\text{prev}}, q_{\text{curr}}, n_{\text{curr}}, W, d] = [0, 0, 0, 0, 1, 0]$
  $s \leftarrow s_0$;
  //*trial*
  **while** $s \notin G$ **do**
    $V_u(s) = \min_{a \in A} \sum_{s' \in S} p(s, a, s')[C(s, a) + V_u(s')]$;
    $V_l(s) = \min_{a \in A} \sum_{s' \in S} p(s, a, s')[C(s, a) + V_l(s')]$;
    $a^* = \pi_p^*(s) = \arg\min_{a \in A} V_l(s)$;
    $\delta = Res^{V_l}(s)$;//$\delta$: *residual of* $V_l(s)$
    $q = \delta W$;//$q$: *update quality score*
    $\Delta(s) = V_u(s) - V_l(s) - \epsilon/2$;//$\Delta(s)$: *excess uncertainty of state* $s$
    $prio(s) = \min\{\Delta(s), \max_{s' \in S}[p(s, a^*, s')prio(s')]\}$;//$prio(s)$:*priority of* $s$
    $s' = \arg\max_{s' \in S}[p(s, a^*, s')prio(s')]$;//*deterministically select* $s'$
    **if** $d > D/k_D$ **then** $[q_{\text{curr}}, n_{\text{curr}}] = [q_{\text{curr}} + q, n_{\text{curr}} + 1]$;
    **else** $[q_{\text{prev}}, n_{\text{prev}}] = [q_{\text{prev}} + q, n_{\text{prev}} + 1]$;
    **if** $(\Delta(s) \leqslant 0) \vee (d \geqslant D)$ **then** break;
    **else**
      $W = p(s, a^*, s')W$;//*W:occupancy of states* visited *by current policy*
      $s \leftarrow s'$;
      $d = d + 1$;
  **if** $q_{\text{curr}}/n_{\text{curr}} \geqslant q_{\text{prev}}/n_{\text{prev}}$ **then** $D = k_D D$//*update* $D$;
return $\pi_p^*(s_0)$;

For a certain state $s' \in S$, we need to know the extent to which an update of $s'$ improves the policy. Assume that the value of $V^*(s')$ is known, then substitute $V^*(s')$ for $[V_l(s') + V_u(s')]/2$ in (4.2):

$$\mathbb{E}[Q(s, a)|\overrightarrow{V}, V^*(s')] = C(s, a) + p(s, a, s')V^*(s') + \sum_{s'' \neq s'} p(s, a, s'')\frac{V_l(s'') + V_u(s'')}{2}. \tag{4.3}$$

Let $a^*$ be the current optimal action for state $s$, then if $a^*$ is replaced by another action $a$ and we know the exact value of $V^*(s')$, the possible reduction of Q-value can be calculated by:

$$\Delta Q[V^*(s')|s, a, a^*, s'] = \max\{0, \mathbb{E}[Q(s, a^*)|\overrightarrow{V}, V^*(s')] - \mathbb{E}[Q(s, a)|\overrightarrow{V}, V^*(s')]\}. \tag{4.4}$$

However in reality, $V^*(s')$ is unknown, we can estimate the expectation of reduction in Q-value thanks to the assumption that $V^*(s') \sim U[V_l(s'), V_u(s')]$, then the VPI value of the state $s'$ is

$$\begin{aligned}\text{VPI}(s'|s, a^*) &= \max_{a \neq a^*} \int_{-\infty}^{+\infty} f[V^*(s')|\overrightarrow{V}]\Delta Q[V^*(s')|s, a, a^*, s']\mathrm{d}V^*(s') \\ &= \begin{cases} \dfrac{1}{V_u(s') - V_l(s')} \max\limits_{a \neq a^*} \displaystyle\int_{V_l(s')}^{V_u(s')} \Delta Q[V^*(s')|s, a, a^*, s']\mathrm{d}V^*(s'), & s' \in S' \\ 0, & s' \in G \cup D \end{cases}. \end{aligned} \tag{4.5}$$

**Algorithm 6:** Adapted VPI–RTDP for SSPADE Model.

$\forall s \in D$, $V_l(s) = \infty$, $V_u(s) = \infty$;
$\forall s \in G$, $V_l(s) = 0$, $V_u(s) = 0$;
$\forall s \in S'$, $V_l(s) = 0$, $V_u(s) = V_u^0$;
**while** *there is time left* **do**
  $s \leftarrow s_0$;
  $[convergence, depth] = [0, 0]$;
  *//trial*
  **while** $(depth \leqslant max\_depth) \wedge (there\ is\ time\ left)$ **do**
    $depth = depth + 1$;
    $V_u(s) = \min_{a \in A} \sum_{s' \in S} p(s, a, s')[C(s, a) + V_u(s')]$;
    $V_l(s) = \min_{a \in A} \sum_{s' \in S} p(s, a, s')[C(s, a) + V_l(s')]$;
    $a^* = \pi_p^*(s) = \arg\min_{a \in A} V_l(s)$;
    $\forall s' \in S'$, $b(s') = p(s, a^*, s')[V_u(s') - V_l(s')]$;
    **if** $s = s_0$ **then** $B = \max_{s'} b(s')$;
    **else** $B = \eta \max_{s'} b(s')$;
    **if** $B > \beta$ **then**
      choose $s'$ from distribution $b(s')/\sum_{s'} b(s')$;
    **else**
      $\forall s' \in S'$, $v(s') = \mathrm{VPI}(s'|s, a^*)$;
      **if** $\max_{s'} v(s') \geqslant \epsilon$ **then**
        choose $s'$ from distribution $v(s')/\sum_{s'} v(s')$;
      **else if** $(\sum_{s'} b(s') \geqslant \epsilon) \wedge (r \sim U(0, 1) < \alpha)$ **then**
        choose $s'$ from distribution $b(s')/\sum_{s'} b(s')$;
      **else**
        **if** $s = s_0$ **then** $convergence = 1$;
        break;
  **if** $convergence = 1$ **then** break;
return $\pi_p^*(s_0)$;

The adapted version of VPI–RTDP is given in Algorithm 6. When some successors of $\{s, a^*\}$ are still far away from being converged, $s'$ is selected in the way of BRTDP. Otherwise, if the gaps between two bounds of all the successors are below $\beta$, the VPI value of every successor is calculated: if the largest VPI value exceeds $\epsilon$, $s'$ is sampled with the probability proportional to the VPI values; if not, VPI–RTDP stops the *trial* or turns back to the same way of selecting $s'$ as BRTDP does.

## 5. Numerical experiments

In this section, we firstly compare the computational efficiencies of the algorithms presented in Sections 3.3 and 4. These algorithms are utilized to solve the same OR planning problem which is formulated as an SSPADE model. Then for a larger-scale OR planning problem, we evaluate the performances of the policies obtained from the SSPADE model with different dead end settings. All the codes are written in C language. The programmes are executed on a PC with an Inter(R) Core(TM) i7-3770 CPU @3.40 GHz and a memory of 8 GB. Notations of performance measures used in this section are given in Table 2.

TABLE 2. Notations of performance measures.

| Symbol | Description |
|---|---|
| $D^u_{\max}$ | Maximum waiting time of patients at urgency level $u$ (day) |
| $D^u_{\text{mean}}$ | Average waiting time of patients at urgency level $u$ (day) |
| $\sigma(D^u)$ | Standard deviation of waiting time of patients at urgency level $u$ |
| $H_o$ | Average over-utilization of ORs per month (hour) |
| $\sigma(H_o)$ | Standard deviation of monthly over-utilization of ORs |
| $\Theta^u_{\text{mean}}$ | Average throughput of patients at urgency level $u$ per month |
| $\sigma(\Theta^u)$ | Standard deviation of monthly throughput of patients at urgency level $u$ |
| $C_{\text{mean}}$ | Average cost per month |
| $\sigma(C)$ | Standard deviation of monthly cost |
| $t_{\text{total}}$ | Total computation time (ms) |
| $t_{\max}$ | Maximum computation time per day (ms) |
| $t_{\text{mean}}$ | Average computation time per day (ms) |
| $\sigma(t)$ | Standard deviation of daily computation time |
| $S_{\text{visited}}$ | Total number of states visited by the agent |
| $N_{\text{MC}}$ | Total number of Monte-Carlo simulations carried out |

### 5.1. Computational efficiencies of the algorithms

We consider an OR planning problem in a small-scale department. One decision period $\tau$ corresponds to one day. The regular capacity of the department is $H = 8$ OR-hours each day. The highest urgency level of the elective patients is $U = 2$. Maximum waiting times for the elective patients are $[T_1, T_2] = [7, 5]$. Let $\theta_u$ be the average arriving rate of the elective patients at level $u$ and $\theta_0$ be the one of the non-electives, and $[\theta_0, \theta_1, \theta_2] = [2, 1, 2]$. Surgery durations are difficult to predict since they depend on various complex factors, *e.g.*, the characteristics of the patient, the surgeon and the surgical team [8]. In literature, most of the relative researches define different types or parameters of probability distributions for the surgery durations of elective patients and emergency patients [9, 37, 50, 51]. For elective patients, Min and Yih [6], Lamiri *et al.* [9], Min and Yih [13] and Truong [14] assume that their surgery durations are identically distributed. Nevertheless, Olivares *et al.* [52] show that surgery durations are not identically distributed and are influenced by patient characteristics and case severities. In addition, Riise *et al.* [38] suggest that surgery durations depend on the surgery procedure, the health and age of the patient, and on the chosen mode. Samudra *et al.* [8] emphasize the importance to separate the patient population when the characteristics of the considered patients are not homogeneous, so that the variability can be reduced. In our work, the patients in the waiting list are classified by urgency levels. As the procedure and the complexity of surgeries for patients at different urgency levels are likely to be different, it is reasonable to assign different distribution parameters to different urgency levels. Let $\mu_0$ and $\mu_u$ be the average surgery durations of the emergency patients and the elective patients at urgency level $u$, respectively, and let $\sigma_0$ and $\sigma_u$ be the corresponding standard deviations. Due to the lack of hospital data, we arbitrarily define $[\mu_0, \mu_1, \mu_2] = [1.5, 1, 2]h$ and $[\sigma_0^2, \sigma_1^2, \sigma_2^2] = [2, 1, 1]h^2$. Unit waiting cost is $c_w = 50$. Over-utilization cost of ORs is $c_o = 350$ per hour.

Considering the department scale and following the principals (3.13) and (3.14), the dead end parameters are determined as follows:

$$\begin{cases} \{l_1(1), \ldots, l_1(7)\} = \{3, 3, 2, 1, 1, 1, 1\} \\ \{l_2(1), \ldots, l_2(5)\} = \{4, 4, 3, 2, 1\} \\ \{L_1, L_2\} = \{5, 5\} \end{cases}. \tag{5.1}$$

Numerical results of different algorithms are demonstrated in Tables 3 and 4. These data are obtained from a simulation of 120 consecutive months (every month contains 30 days). Stochastic variables including arrivals of patients and surgery durations are generated by Monte-Carlo simulation. In each algorithm except for RTDP,

TABLE 3. Computational efficiencies of different algorithms.

| Algorithm | Type | $t_{\text{total}}$ | $t_{\max}$ | $t_{\text{mean}}$ | $\sigma(t)$ |
|---|---|---|---|---|---|
| VI | Off line | 1 849 693 | – | – | – |
| RTDP | On line | 910 678 | 329 | 252.966 | 46.165 |
| LRTDP | On line | 7395 | 2 491 | 2.054 | 43.510 |
| BRTDP | On line | 24 812 | 24 016 | 6.892 | 400.223 |
| FRTDP | On line | 31 494 | 31 285 | 8.748 | 521.343 |
| VPI–RTDP | On line | 20 389 | 11 962 | 5.664 | 208.343 |

**Notes.** The notations used in this table are defined in Table 2.

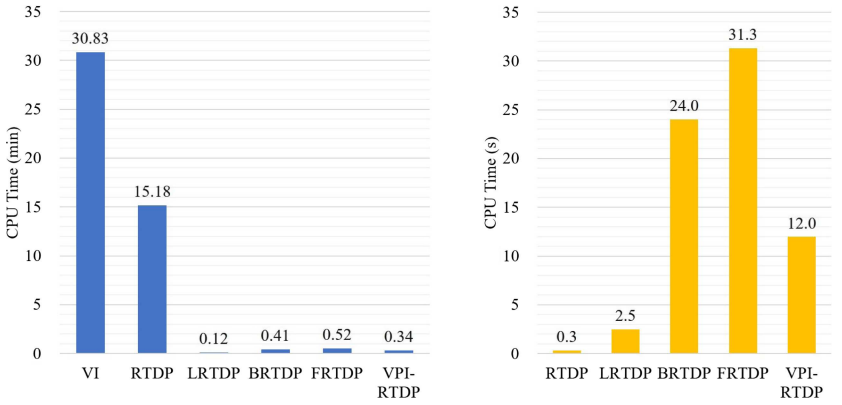
TABLE 4. Simulation results of different algorithms.

| Algorithm | $D^1_{\max}$ | $D^2_{\max}$ | $D^1_{\text{mean}}$ | $\sigma(D^1)$ | $D^2_{\text{mean}}$ | $\sigma(D^2)$ | $H_o$ | $\sigma(H_o)$ | $\Theta^1_{\text{mean}}$ | $\sigma(\Theta^1)$ | $\Theta^2_{\text{mean}}$ | $\sigma(\Theta^2)$ | $C_{\text{mean}}$ | $\sigma(C)$ | $N_{\text{MC}}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VI | 4 | 2 | 1.55 | 0.15 | 1.22 | 0.05 | 42.34 | 13.54 | 29.57 | 4.67 | 57.49 | 6.81 | 17 018 | 5022 | 28 447 |
| RTDP | 7 | 4 | 1.53 | 0.16 | 1.22 | 0.05 | 42.78 | 14.71 | 29.18 | 5.30 | 57.25 | 6.47 | 17 138 | 5448 | 28 370 |
| LRTDP | 7 | 5 | 1.67 | 0.23 | 1.21 | 0.05 | 41.65 | 14.93 | 28.96 | 4.43 | 57.35 | 6.79 | 17 259 | 5563 | 28 356 |
| BRTDP | 5 | 2 | 1.54 | 0.16 | 1.22 | 0.05 | 44.26 | 15.10 | 29.58 | 5.26 | 57.85 | 7.02 | 17 675 | 5587 | 28 497 |
| FRTDP | 7 | 5 | 1.65 | 0.17 | 1.24 | 0.05 | 42.49 | 14.89 | 29.33 | 5.23 | 57.33 | 6.87 | 17 510 | 5577 | 28 389 |
| VPI–RTDP | 4 | 2 | 1.52 | 0.16 | 1.22 | 0.05 | 43.41 | 15.29 | 29.43 | 4.82 | 57.42 | 6.96 | 17 330 | 5622 | 28 422 |

**Notes.** The notations used in this table are defined in Table 2.

the threshold of convergence is $\epsilon = 1$; in RTDP, the number and the depth of trials are $N = max\_depth = 20$; in BRTDP, $\eta = 1.1$; in FRTDP, the initial depth of trial is $D_0 = 100$ and the coefficient $k_D$ is 1.1; in VPI–RTDP, the parameters are $\alpha = 0.01$, $\beta = 15$, $\eta = 1$, $max\_depth = 1000$, and the maximum computation time per day is $\infty$. Lower and upper bounds are initialized as 0 and 6000, respectively.

It can be seen from Table 3 that VI is the most time-consuming algorithm: nearly 31 min are used, while all the RTDP-based algorithms spend much less time. Table 4 shows that the policy obtained by VI allows the patients at urgency levels 1 and 2 to wait for no longer than 4 and 2 days, respectively. Compared to VI, RTDP reduces the computation time by more than 50% and LRTDP consumes the least CPU time (less than 8 s), but Table 4 shows that these two algorithms significantly increase the maximum waiting times of patients. These policy deteriorations result from the following facts: RTDP could waste time on visiting the states with converged value functions as it lacks a convergence detection, while some of the states are still not converged when the maximum number of trials are reached; LRTDP provides few quality guarantees and a policy obtained by maintaining only a lower bound could be arbitrarily bad [49]. BRTDP and FRTDP are two similar algorithms as they both maintain double bounds on the optimal value function and converge faster than the original RTDP, whereas the computation time spent by BRTDP is 21% less than that spent by FRTDP. Moreover, FRTDP leads to a sharper increase in the maximum waiting time of patients than BRTDP does. Finally, VPI–RTDP converges even faster than BRTDP: CPU time consumed by the former is 17.8% less than that consumed by the latter. This is due to the fact that VPI–RTDP saves time by terminating the computation as long as the policy is converged. More importantly, VPI–RTDP does not cause any prolongation of patient waiting time. Figures 1 and 2 compare the computational efficiencies and main performance measures of these different algorithms. Other than the maximum waiting time, no significant difference is found in the other performance measures. We can thereby conclude that the adapted VPI–RTDP algorithm (presented in Algorithm 6) is the most effective approach to solve the proposed SSPADE model for OR planning.




(a) Total Computation Time (b) Max Computation Time per Day

FIGURE 1. Computation times of different algorithms.

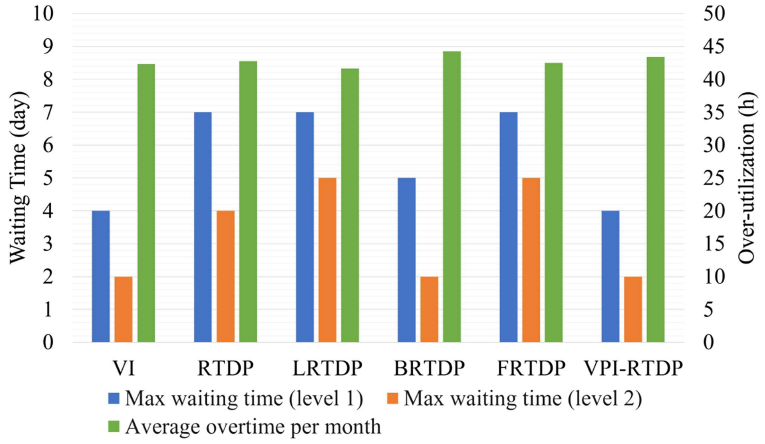



FIGURE 2. Main performance measures of different algorithms.

## 5.2. Performance with different dead end settings

In Section 5.1, we solve a small-scale OR planning problem to compare the effectiveness of different algorithms. In this section, we deal with a larger problem considering the scale of real cases. Given that the memory and the CPU time needed for solving the problem increase exponentially as the problem scale becomes larger, the algorithms which occupy too much memory or converge too slowly like VI are unable to solve large-scale problems. It is analyzed (see Sect. 4.3) and proved through numerical experiments (see Sect. 5.1) that VPI–RTDP is the most proper method to solve the proposed OR planning model. To verify its capability of solving larger-scale problems, we modify the model parameters in Section 5.1 and adopt VPI–RTDP to compute the policy. The regular OR capacity is increased to $H = 12$ h per day. Maximum allowed waiting times are modified as $[T_1, T_2] = [20, 10]$. Arriving rates of the elective patients are $[\theta_0, \theta_1, \theta_2] = [2, 2, 3]$. The other parameters remain the same. Due to the lack of hospital data, we refer to related works [6,13] when determining these parameters. Min and Yih [6,13] deal with the scheduling of one type of surgeries (coronary artery bypass grafting surgeries) with consideration of single resource: the OR capacity. The same problem setting is adopted in this paper. As for parameter settings, the studied problems in [6,13] are based on real scenarios. In [6], patients are divided into

TABLE 5. Dead end settings.

| Instance | $L_1$ | $l_1(t)$ | | $L_2$ | $l_2(t)$ | |
|---|---|---|---|---|---|---|
| | | $t=1$ | $2 \leqslant t \leqslant T_1$ | | $t=1$ | $2 \leqslant t \leqslant T_2$ |
| M0 | 8 | 4 | $\{4,3,2,1,1,\ldots,1\}$ | 10 | 6 | $\{6,5,4,3,2,1,1,\ldots,1\}$ |
| M1 | 8 | 4 | $\{4,3,2,1,1,\ldots,1\}$ | 10 | 6 | $\{\mathbf{4},\mathbf{2},\mathbf{1},\mathbf{1},\ldots,\mathbf{1}\}$ |
| M2 | 8 | 4 | $\{4,3,2,1,1,\ldots,1\}$ | **8** | 6 | $\{\mathbf{4},\mathbf{2},\mathbf{1},\mathbf{1},\ldots,\mathbf{1}\}$ |
| M3 | **6** | 4 | $\{4,3,2,1,1,\ldots,1\}$ | 10 | 6 | $\{6,5,4,3,2,1,1,\ldots,1\}$ |
| M4 | **6** | 4 | $\{\mathbf{2},\mathbf{1},\mathbf{1},\ldots,\mathbf{1}\}$ | 10 | 6 | $\{6,5,4,3,2,1,1,\ldots,1\}$ |
| M5 | **6** | 4 | $\{\mathbf{2},\mathbf{1},\mathbf{1},\ldots,\mathbf{1}\}$ | **8** | 6 | $\{\mathbf{4},\mathbf{2},\mathbf{1},\mathbf{1},\ldots,\mathbf{1}\}$ |
| M6 | **5** | 4 | $\{\mathbf{2},\mathbf{1},\mathbf{1},\ldots,\mathbf{1}\}$ | **7** | 6 | $\{\mathbf{4},\mathbf{2},\mathbf{1},\mathbf{1},\ldots,\mathbf{1}\}$ |

TABLE 6. Computational efficiencies in different dead end settings.

| Instance | $t_{\text{total}}$ | $t_{\max}$ | $t_{\text{mean}}$ | $\sigma(t)$ | $S_{\text{visited}}$ |
|---|---|---|---|---|---|
| M0 | 750 284 679 | 14 400 000 | 694 708 | 2 987 042 | 136 077 |
| M1 | 469 948 783 | 14 400 000 | 435 138 | 2 318 980 | 107 853 |
| M2 | 34 613 013 | 8 073 408 | 32 049 | 324 223 | 28 346 |
| M3 | 24 911 429 | 5 453 318 | 23 066 | 271 383 | 25 973 |
| M4 | 23 757 212 | 4 281 757 | 22 923 | 212 718 | 24 980 |
| M5 | 3 464 677 | 505 948 | 3208 | 25 740 | 5116 |
| M6 | 852 227 | 140 715 | 789 | 4 380 | 858 |

**Notes.** The notations used in this table are defined in Table 2.

three urgency groups, and new arrivals are modeled as Poisson distribution with parameters 1, 5, 3, respectively. Though the authors do not specify the time length of a period, they mention that patients are scheduled a week or a few days in advance. Hence we can speculate that one decision period corresponds to about one week, thus the average number of new arrivals per week in their studied problem is 9 in total. In addition, the regular OR capacity in [6] is determined as 480 min (8 h) per week. In [13], there are two urgency levels with arrival rates equal to 6 and 7 per week, respectively, and the regular OR capacity is 780 min (13 h) per week. In comparison, we consider three urgency levels of patients (emergency patients can be regarded as the highest urgency level) with weekly arrival rates of new patients equal to 49 (or 35 if weekends are not considered) in total. The regular OR capacity is 84 h (or 60 h if weekends are not considered) per week. It can be seen that the problem scale studied in our work is larger than that in [6,13]. Therefore, if the adapted VPI–RTDP algorithm can effectively solve the cases in this section, it is also capable of solving real cases. We try different dead end settings, shown in Table 5, to analyze their effects on the policy and the computational efficiency. For a group of patients at a certain urgency level $u$, there are two dead end parameters: $L_u$ and $l_u(t)$. In Table 5, M0 is the basic instance, and the changes of $L_u$ and $l_u(t)$ in instances M1–M6 are emphasized by bold text.

VPI–RTDP is adopted to solve the instances shown in Table 5. Parameters of the algorithm are set as follows: $\alpha = 0.01$, $\beta = 10$, $\epsilon = 1$, $\eta = 10$, $max\_depth = 1500$ and the maximum computation time per day is 4 hours. Experimental results obtained from a simulation of 36 consecutive months are presented in Tables 6 and 7.

It can be seen from Table 6 that the total CPU time and the number of visited states dramatically decrease as the dead end parameters $L_u$ and $l_u(t)$ are reduced. The reason is that the computations are restricted in subset $S'$ whose border is determined by $L_u$ and $l_u(t)$. Figure 3 provides an intuitive comparison of the computational efficiencies in different instances. Instance M0 consumes the most CPU time, *i.e.*, more than 208 h in total, whereas in M1 where only $l_2(t)$ is reduced, $t_{\text{total}}$ and $S_{\text{visited}}$ decrease by 37.4% and 20.7%, respectively. Further, in M2 where both $L_2$ and $l_2(t)$ are reduced, only 4.6% of the computation time of M0 is consumed and $S_{\text{visited}}$ is reduced by 79.2%. The reductions of $t_{\text{total}}$ and $S_{\text{visited}}$ are more obvious in M5 and

TABLE 7. Simulation results of different dead end settings.

| Instance | $D^1_{\max}$ | $D^2_{\max}$ | $D^1_{\text{mean}}$ | $\sigma(D^1)$ | $D^2_{\text{mean}}$ | $\sigma(D^2)$ | $H_o$ | $\sigma(H_o)$ | $\Theta^1_{\text{mean}}$ | $\sigma(\Theta^1)$ | $\Theta^2_{\text{mean}}$ | $\sigma(\Theta^2)$ | $C_{\text{mean}}$ | $\sigma(C)$ | $N_{\text{MC}}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M0 | 10 | 8 | 1.50 | 0.18 | 1.30 | 0.15 | 35.28 | 15.41 | 56.14 | 7.61 | 89.36 | 9.78 | 17 080 | 6678 | 10 640 |
| M1 | 8 | 5 | 1.45 | 0.20 | 1.28 | 0.14 | 35.06 | 16.43 | 58.75 | 5.20 | 88.47 | 8.11 | 16 973 | 8689 | 10 702 |
| M2 | 5 | 2 | 1.47 | 0.19 | 1.19 | 0.08 | 34.51 | 15.69 | 59.64 | 6.00 | 88.72 | 10.94 | 16 478 | 6208 | 10 732 |
| M3 | 4 | 3 | 1.37 | 0.12 | 1.35 | 0.17 | 34.91 | 14.54 | 57.03 | 6.31 | 91.69 | 9.40 | 16 819 | 5783 | 10 748 |
| M4 | 3 | 3 | 1.34 | 0.12 | 1.30 | 0.17 | 35.01 | 14.21 | 57.61 | 6.52 | 88.08 | 9.24 | 16 146 | 6444 | 10 656 |
| M5 | 4 | 2 | 1.33 | 0.10 | 1.20 | 0.07 | 37.56 | 12.94 | 57.36 | 5.35 | 89.47 | 8.90 | 15 971 | 5743 | 10 687 |
| M6 | 3 | 2 | 1.16 | 0.05 | 1.10 | 0.03 | 38.46 | 14.56 | 57.39 | 7.17 | 88.19 | 8.26 | 15 881 | 5389 | 10 681 |

**Notes.** The notations used in this table are defined in Table 2.

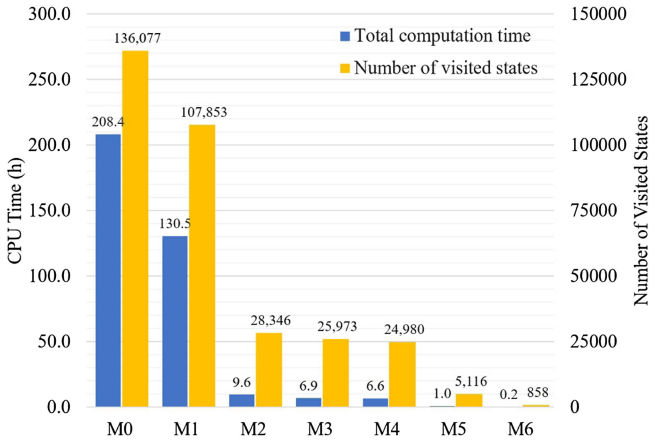


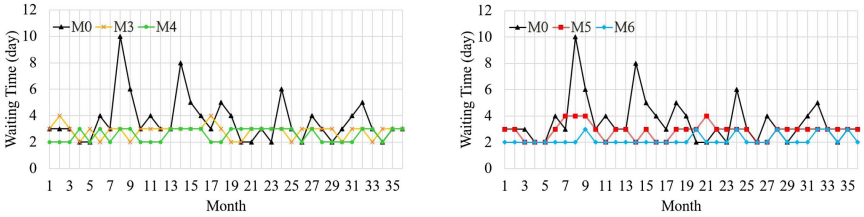

FIGURE 3. Computational efficiencies in different dead end settings.

M6 where dead end parameters $L_u$ and $l_u(t)$ for all levels of patients are reduced. Especially in M6, $t_{\text{total}}$ and $S_{\text{visited}}$ are reduced by 99.9% and 99.4%, respectively in comparison with M0.

Table 7 shows that the waiting times of patients at urgency level 2 in instances M1 and M2 are obviously lower compared to M0. Specifically, $D^2_{\max}$ is shortened from 8 days to 5 and 2 days, respectively and $D^2_{\text{mean}}$ is reduced by 1.5% and 8.5%, respectively. In the meanwhile, there is no increase in $D^1_{\max}$ and $D^1_{\text{mean}}$. Similarly, the waiting times of patients at urgency level 1 are shortened in instances M3 and M4. In M5 and M6, all levels of patients wait for less time than in M0. Figures 4 and 5 compare the evolutions of patient waiting times over the 36 months in different instances. The curves in Figures 4 and 5 indicate that the more $L_u$ and/or $l_u(t)$ are reduced, the less the patients at related urgency level wait. Therefore, we can conclude that reducing the dead end parameters $L_u$ and/or $l_u(t)$ helps to shorten the patient waiting times.

However, reducing $L_u$ and/or $l_u(t)$ may also lead to more over-utilization of ORs especially when the dead end parameters are reduced too much. In instances M2–M6, $H_o$ gradually increases as $L_u$ and $l_u(t)$ decrease. For example, the overtime of OR in M6 is 11.4% more than that in M2. Moreover, the OR over-utilization in instances M0 and M1 are higher than that in M2–M4, since the policies of some states are still not converged to the optimum when the daily computation time reaches 4 h (as shown in Tab. 6). Despite of the increase in the overuse of ORs, the total cost shows a downtrend as $L_u$ and $l_u(t)$ decrease. Because the reduction of patient waiting cost is larger than the growth of OR over-utilization cost.

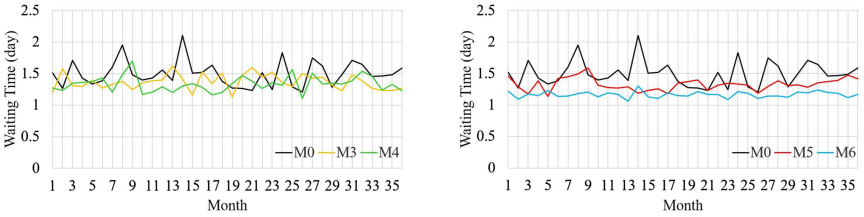
Based on the analyses above, we can conclude that if the studied problem is formulated as a regular MDP model where dead ends are not incorporated, *i.e.*, $\forall u \in [1, U]$ and $\forall t \in [1, T_u]$: $L_u = l_u(t) = \infty$, the




(a) urgency level $u = 1$ (b) urgency level $u = 1$

(c) urgency level $u = 2$ (d) urgency level $u = 2$

FIGURE 4. Comparison of maximum patient waiting times.




(a) urgency level $u = 1$ (b) urgency level $u = 1$

(c) urgency level $u = 2$ (d) urgency level $u = 2$

FIGURE 5. Comparison of average patient waiting times.

computational efficiency will be much lower (or the problem will be unsolvable due to the infinite state space) and elective patients will wait for much longer time before being served.

## 6. CONCLUSION

In this work we propose an SSPADE formulation for OR planning which aims at shortening the waiting times of elective patients and controlling the over-utilization of ORs. Elective patients and non-elective ones (emergencies) are both considered in our model, and they share the same ORs to guarantee that the emergencies can be served within the shortest delay. A time-dependent prioritization scoring system is adopted to prevent the electives from staying in the waiting list for too long time. Different from a conventional SSP MDP model,

we incorporate dead ends into the decision-making process to improve the policy as well as save CPU time. Given that VI is insufficient to solve many of the realistic problems with large state space and action space, we adapt the RTDP algorithm and its variations to solve the proposed model.

Numerical experiments are carried out to test the adapted RTDP-based algorithms and the proposed model. Firstly, we apply different algorithms to solve the same SSPADE model and compare the simulation results. VI needs more than 30 min to provide an optimal policy, whereas RTDP consumes less than 16 min. Compared with RTDP, its variations save even much more CPU time as they are able to finish the computations within one minute. Among these algorithms, VPI–RTDP demonstrates the highest performance since it consumes only 18 s of CPU time and provides a policy which is as good as the one provided by VI. Secondly, we use VPI–RTDP to solve a larger-scale OR planning problem formulated as SSPADE. Different combinations of dead end parameters ($L_u$ and $l_u(t)$) are evaluated and simulation results reveal that incorporating dead ends into the model contribute to the improvement of computational efficiency and the shortening of patient waiting times. However, over-utilizations of ORs may significantly increase if the dead end parameters are reduced too much.

There are several potential extensions for the future researches. Firstly we will conduct simulations with statistic data from hospitals to evaluate the effectiveness and the practicability of the SSPADE model in real world. Secondly, RTDP-based algorithms might not be efficient enough for realistic instances with larger number of patients. Thus developing approximate dynamic programming (ADP) methods with higher computational efficiencies might be necessary. Moreover, the OR planning model based on SSPADE presented in this work could be extended to a more comprehensive one by taking more medical resources into consideration. We hope to integrate the planning of ORs with the management of other upstream and/or downstream facilities in our future works.

## References


[1] F. Sperandio, C. Gomes, J. Borges, A.C. Brito and B. Almada-Lobo, An intelligent decision support system for the operating theater: a case study. *IEEE Trans. Autom. Sci. Eng.* **11** (2014) 265–273.

[2] Y. Wang, J. Tang, Z. Pan and C. Yan, Particle swarm optimization-based planning and scheduling for a laminar-flow operating room with downstream resources. *Soft Comput.* **19** (2015) 2913–2926.

[3] R. Aringhieri, P. Landa, P. Soriano, E. Tànfani and A. Testi, A two level metaheuristic for the operating room scheduling and assignment problem. *Comput. Oper. Res.* **54** (2015) 21–34.

[4] C. Van Riet and E. Demeulemeester, Trade-offs in operating room planning for electives and emergencies: a review. *Oper. Res. Health Care* **7** (2015) 52–69.

[5] S. Zhu, W. Fan, S. Yang, J. Pei and P.M. Pardalos, Operating room planning and surgical case scheduling: a review of literature. *J. Comb. Optim.* (2018) 1–49.

[6] D. Min and Y. Yih, An elective surgery scheduling problem considering patient priority. *Comput. Oper. Res.* **37** (2010) 1091–1099.

[7] B. Addis, G. Carello, A. Grosso and E. Tànfani, Operating room scheduling and rescheduling: a rolling horizon approach. *Flexible Serv. Manuf. J.* **28** (2016) 206–232.

[8] M. Samudra, C. Van Riet, E. Demeulemeester, B. Cardoen, N. Vansteenkiste and F.E. Rademakers, Scheduling operating rooms: achievements, challenges and pitfalls. *J. Scheduling* **19** (2016) 493–525.

[9] M. Lamiri, X. Xie, A. Dolgui and F. Grimaud, A stochastic model for operating room planning with elective and emergency demand for surgery. *Eur. J. Oper. Res.* **185** (2008) 1026–1037.

[10] Y. Ferrand, M. Magazine and U. Rao, Comparing two operating-room-allocation policies for elective and emergency surgeries. In: *Proceedings of the 2010 Winter Simulation Conference*. IEEE (2010) 2364–2374.

[11] B. Cardoen, E. Demeulemeester and J. Beliën, Operating room planning and scheduling: a literature review. *Eur. J. Oper. Res.* **201** (2010) 921–932.

[12] F. Guerriero and R. Guido, Operational research in the management of the operating theatre: a survey. *Health Care Manage. Sci.* **14** (2011) 89–114.

[13] D. Min and Y. Yih, Managing a patient waiting list with time-dependent priority and adverse events. *RAIRO: OR* **48** (2014) 53–74.

[14] V.-A. Truong, Optimal advance scheduling. *Manage. Sci.* **61** (2015) 1584–1597.

[15] H. Saadouli, B. Jerbi, A. Dammak, L. Masmoudi and A. Bouaziz, A stochastic optimization and simulation approach for scheduling operating rooms and recovery beds in an orthopedic surgery department. *Comput. Ind. Eng.* **80** (2015) 72–79.

[16] I. Marques and M.E. Captivo, Different stakeholders' perspectives for a surgical case assignment problem: deterministic and robust approaches. *Eur. J. Oper. Res.* **261** (2017) 260–278.

[17] R.L. Burdett and E. Kozan, An integrated approach for scheduling health care activities in a hospital. *Eur. J. Oper. Res.* **264** (2018) 756–773.

[18] G. Latorre-Núñez, A. Lüer-Villagra, V. Marianov, C. Obreque, F. Ramis and L. Neriz, Scheduling operating rooms with consideration of all resources, post anesthesia beds and emergency surgeries. *Comput. Ind. Eng.* **97** (2016) 248–257.

[19] M. Heydari and A. Soudi, Predictive/reactive planning and scheduling of a surgical suite with emergency patient arrival. *J. Med. Syst.* **40** (2016) 30.

[20] Y. Wang, J. Tang and R.Y. Fung, A column-generation-based heuristic algorithm for solving operating theater planning problem under stochastic demand and surgery cancellation risk. *Int. J. Prod. Econ.* **158** (2014) 28–36.

[21] S.H. Hashemi Doulabi, L.-M. Rousseau and G. Pesant, A constraint-programming-based branch-and-price-and-cut approach for operating room planning and scheduling. *INFORMS J. Comput.* **28** (2016) 432–448.

[22] P. Landa, R. Aringhieri, P. Soriano, E. Tànfani and A. Testi, A hybrid optimization algorithm for surgeries scheduling. *Oper. Res. Health Care* **8** (2016) 103–114.

[23] S. Neyshabouri and B.P. Berg, Two-stage robust optimization approach to elective surgery and downstream capacity planning. *Eur. J. Oper. Res.* **260** (2017) 21–40.

[24] J. Patrick, M.L. Puterman and M. Queyranne, Dynamic multipriority patient scheduling for a diagnostic resource. *Oper. Res.* **56** (2008) 1507–1525.

[25] M.E. Zonderland, R.J. Boucherie, N. Litvak and C.L. Vleggeert-Lankamp, Planning and scheduling of semi-urgent surgeries. *Health Care Manage. Sci.* **13** (2010) 256–267.

[26] N. Hosseini and K. Taaffe, Evaluation of optimal scheduling policy for accommodating elective and non-elective surgery via simulation. In: *Proceedings of the 2014 Winter Simulation Conference*. IEEE Press (2014) 1377–1386.

[27] D. Astaraky and J. Patrick, A simulation based approximate dynamic programming approach to multi-class, multi-resource surgical scheduling. *Eur. J. Oper. Res.* **245** (2015) 309–319.

[28] A. Kolobov, Mausam and D.S. Weld, A theory of goal-oriented MDPs with dead ends. In: *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*. UAI'12 (2012) 438–447.

[29] A. Kolobov, Mausam and D.S. Weld, Stochastic shortest path MDPs with dead ends. In: *ICAPS Heuristics and Search for Domain Independent Planning (HSDIP) Workshop* (2012).

[30] M. Heng and J.G. Wright, Dedicated operating room for emergency surgery improves access and efficiency. *Can. J. Surgery* **56** (2013) 167.

[31] A. Leppäniemi and I. Jousela, A traffic-light coding system to organize emergency surgery across surgical disciplines. *Br. J. Surgery* **101** (2014) e134–e140.

[32] I. Adan, J. Bekkers, N. Dellaert, J. Jeunet and J. Vissers, Improving operational effectiveness of tactical master plans for emergency and elective patients under stochastic demand and capacitated resources. *Eur. J. Oper. Res.* **213** (2011) 290–308.

[33] J.T. van Essen, E.W. Hans, J.L. Hurink and A. Oversberg, Minimizing the waiting time for emergency surgery. *Oper. Res. Health Care* **1** (2012) 34–44.

[34] J.-S. Tancrez, B. Roland, J.-P. Cordier and F. Riane, Assessing the impact of stochasticity for operating theater sizing. *Decis. Support Syst.* **55** (2013) 616–628.

[35] A. Testi, E. Tanfani, R. Valente, G. Ansaldo and G. Torre, Prioritizing surgical waiting lists. *J. Eval. Clin. Pract.* **14** (2008) 59–64.

[36] R. Valente, A. Testi, E. Tanfani, M. Fato, I. Porro, M. Santo, G. Santori, G. Torre and G. Ansaldo, A model to prioritize access to elective surgery on the basis of clinical urgency and waiting time. *BMC Health Serv. Res.* **9** (2009) 1.

[37] D. Min and Y. Yih, Scheduling elective surgery under uncertainty and downstream capacity constraints. *Eur. J. Oper. Res.* **206** (2010) 642–652.

[38] A. Riise, C. Mannino and E.K. Burke, Modelling and solving generalised operational surgery scheduling problems. *Comput. Oper. Res.* 66 (2016) 1–11.

[39] P. Punnakitikashem, J.M. Rosenberger and D.B. Behan, Stochastic programming for nurse assignment. *Comput. Optim. App.* **40** (2008) 321–349.

[40] M. Holte and C. Mannino, The implementor/adversary algorithm for the cyclic and robust scheduling problem in health-care. *Eur. J. Oper. Res.* **226** (2013) 551–559.

[41] A.G. Barto, S.J. Bradtke and S.P. Singh, Learning to act using real-time dynamic programming. *Artif. Intell.* **72** (1995) 81–138.

[42] B. Bonet and H. Geffner, Labeled RTDP: improving the convergence of real-time dynamic programming. In: *Proceedings of Thirteenth International Conference on Automated Planning and Scheduling* **3** (2003) 12–21.

[43] H.B. McMahan, M. Likhachev and G.J. Gordon, Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In: *Proceedings of the 22nd International Conference on Machine Learning*. ACM (2005) 569–576.

[44] T. Smith and R. Simmons, Focused real-time dynamic programming for MDPs: squeezing more out of a heuristic. In: *Proceedings of the 21st National Conference on Artificial Intelligence*. AAAI'06 **2** (2006) 1227–1232.

[45] S. Sanner, R. Goetschalckx, K. Driessens and G. Shani, Bayesian real-time dynamic programming. In: *Proceedings of the 21st International Joint Conference on Artifical Intelligence*. IJCAI'09 (2009) 1784–1789.

[46] J. Zhang, M. Dridi and A. El Moudni, A stochastic shortest-path MDP model with dead ends for operating rooms planning. In: *2017 23rd International Conference on Automation and Computing (ICAC)*. IEEE (2017) 1–6.

[47] D.P. Strum, J.H. May, A.R. Sampson, L.G. Vargas and W.E. Spangler, Estimating times of surgeries with two component procedures: comparison of the lognormal and normal models. *Anesthesiol. J. Am. Soc. Anesthesiologists* **98** (2003) 232–240.

[48] G. Xiao, W. van Jaarsveld, M. Dong and J. van de Klundert, Stochastic programming analysis and solutions to schedule overcrowded operating rooms in china. *Comput. Oper. Res.* **74** (2016) 78–91.

[49] Mausam and A. Kolobov, Planning with Markov decision processes: an AI perspective. *Synth. Lect. Artif. Intel. Mach. Learn.* **6** (2012) 1–210.

[50] M. Lamiri, X. Xie and S. Zhang, Column generation approach to operating theater planning with elective and emergency patients. *IIE Trans.* **40** (2008) 838–852.

[51] L. Koppka, L. Wiesche, M. Schacht and B. Werners, Optimal distribution of operating hours over operating rooms using probabilities. *Eur. J. Oper. Res.* **267** (2018) 1156–1171.

[52] M. Olivares, C. Terwiesch and L. Cassorla, Structural estimation of the newsvendor model: an application to reserving operating room time. *Manage. Sci.* **54** (2008) 41–55.