ACCSCIENCE PUBLISHING
An International Journal of Optimization and Control: Theories & Applications
ISSN: 2146-0957 eISSN: 2146-5703
Vol. 16, No. 3, pp.1142-1156 (2026)
https://doi.org/10.36922/IJOCTA025310134
IJOCTA
RESEARCH ARTICLE

Dynamic operating room scheduling with emergency arrivals via deep reinforcement learning

Farshad Ahmadian, Mohammad Bagher Fakhrzad*, Hasan Hosseini-Nasab, and Davood Shishebori

Department of Industrial Engineering, Faculty of Engineering, Yazd University, Yazd, Yazd, Iran
ahmadian.farshad@stu.yazd.ac.ir;mfakhrzad@yazd.ac.ir;hhn@yazd.ac.ir;Shishebori@yazd.ac.ir

ARTICLE INFO

Article history:
Received: July 30, 2025
1st revised: October 30, 2025
2nd revised: November 15, 2025
Accepted: November 26, 2025
Published Online: May 29, 2026

Keywords:
Dynamic operating room scheduling
Deep reinforcement learning
Emergency arrivals
Proximal policy optimization
Markov decision process
Real-time healthcare operations

ABSTRACT

Operating room (OR) scheduling in modern hospitals must reconcile elective case planning precision with emergency arrival uncertainty. Traditional two-phase approaches—offline elective and emergency case scheduling and emergency calling forth rescheduling in the reactive mode—are marred by computational delays and the wasteful use of resources. We introduce an integrated, real-time dynamic OR scheduling model grounded on deep reinforcement learning (DRL). By casting the OR scheduling problem as a Markov decision process, our method continuously adapts surgery assignments, start times, and overtime accruals to both elective and emergent requirements. An actor–critic structure trained with proximal policy optimization learns to progressively dispatch cases to minimize total overtime without separate rescheduling steps. On synthetic benchmarks for small, medium, and large instances with uniform and Poisson emergency-arrival distributions, the DRL outperformed a reactive mixed-integer programming (RMIP) baseline and an optimized heuristic. It reduced average overtime by over 50% while generating schedules in less than 4 s—achieving 100–1,000× speedup over RMIP. These results demonstrate that our learning-based method offers a scalable, reliable, and real-time solution to dynamic OR scheduling, paving the way for deployment in high-stakes clinical environments.

1. Introduction

Contemporary healthcare demands optimal hospital performance while providing quality patient care.[1–3] One of the significant challenges in this field is operating room (OR) scheduling, which is increasingly complicated by the overlap between elective surgeries and emergency admissions.[4–6] Traditional scheduling methods, such as mixed scheduling and metaheuristic algorithms, are typically designed for elective surgeries and are less efficient in the face of unexpected crises.[7–9] This highlights the need for an integrated, real-time scheduling approach that can simultaneously handle both routine and emergency cases.[10–12]

Simulation models used to analyze OR scheduling strategies have limitations in dynamic and uncertain healthcare environments.[13] These models usually rely on predefined decision rules and fixed assumptions and cannot respond to real-time changes.[14] Furthermore, these methods are extremely time-consuming and expensive due to the need to calibrate parameters for each scenario.[15]

In this regard, deep reinforcement learning (DRL) has been proposed as a novel approach that can dynamically learn planning policies through interaction with the environment and respond to uncertainties and changes in real time.[16,17] By integrating the representational power of deep

* Corresponding author.

learning and the sequential decision-making capability of reinforcement learning, DRL can adjust planning decisions in real time and optimize long-term outcomes.

In this study, we proposed a new framework for dynamic operating room scheduling (DORS) using DRL. This method formulates the scheduling problem as a Markov decision process (MDP) and transforms the entire scheduling process into a continuous decision-making process. Using an actor–critic architecture and proximal policy optimization (PPO), our model not only improves decision-making speed but also achieves high-quality scheduling under tight constraints. Our extensive experiments showed that the DRL-based solution significantly reduced overhead and outperformed existing methods in terms of quality and computational efficiency. This advancement can lead to improved patient outcomes and a more efficient use of limited hospital resources. Consequently, this research represents the next step in advancing DORS using DRL and can significantly change how hospitals address operational complexities in dynamic environments.

The rest of the paper is organized as follows: **Section 2** provides a review of relevant literature on DORS and reinforcement learning in healthcare settings. **Section 3** formulates and states the DORS problem mathematically. **Section 4** introduces our DRL architecture and training processes. **Section 5** reports numerical experiments and discusses findings, and **Section 6** concludes with remarks and future research directions.

2. Literature review

In this section, we review the state of the art in OR scheduling under uncertainty by examining (i) key modeling approaches and uncertainty-handling techniques, (ii) traditional exact, metaheuristic, and heuristic solution methods, (iii) their limitations in dynamic, emergency-driven settings, and (iv) emerging DRL methods for healthcare operations.

Recent research suggests that OR scheduling must balance elective and emergency surgeries under uncertainty.[17,18] Several studies have proposed robust and stochastic models for this purpose, including a two-stage model considering random surgery duration and emergency patient arrivals,[4] a robust data-driven model that allocates room capacity for elective and probable emergency patients and uses rolling horizon rescheduling,[19] and the hierarchical framework by Zhao et al.[20] that combines weekly scheduling, daily allocation, and intraday rescheduling using genetic algorithms and particle swarm optimization. These models emphasize the existing consensus on the key role of uncertainty in surgery duration and emergency patient arrivals.[21–23] In classical approaches, mathematical programming models, such as mixed-integer programming (MIP), stochastic programming, and robust optimization, have been used.[4,7] Stochastic programming minimizes expected costs (e.g., overtime, unemployment, surgery cancellations), but its dimension grows rapidly with the number of scenarios;[9,24] in contrast, robust models focus on worst-case scenarios or on uncertain distributions.[7,19] In addition, heuristic and simulation methods are common and include greedy rules, metaheuristic algorithms such as genetic algorithms and forbidden search, and discrete-event simulation to evaluate policies and measure program stability.[2,13,17,25–27] Overall, the literature in this area encompasses a broad set of stochastic, robust, data-driven, simulation, and metaheuristic tools, all focused on simultaneously optimizing patient priority and resource efficiency under uncertainty.[28–34]

Researchers have used a variety of methods to address the OR scheduling problem.[4,6,28] Kamran et al.[9] presented a multi-objective mixed-integer linear programming model for assigning elective patients, with the option to include emergency patients. Subsequently, metaheuristic approaches, such as genetic algorithms, simulated annealing, forbidden search, and ant colony optimization, were used to accelerate the solution of these hybrid models,[26,27] which exhibit higher efficiency and effectiveness compared to simple rules. In practice, many hospitals use heuristic rules or simulation–optimization cycles to generate feasible plans.[2,35] In this approach, the first stage involves a basic selected plan, and the second stage involves replanting after observing actual events.[24,25,36] These models are optimized using sample approximation or gradual algorithms and reduce expected costs such as waiting, idleness, and overtime.[2,20] However, these methods rely on known probability distributions and are vulnerable to rare or uncertain events.[7,13,24]

In contrast, robust optimization offers an alternative by considering the worst-case scenarios of uncertainty. For example, Shehadeh and Padman[7] proposed a robust model aiming to minimize the worst-case expected cost for OR overtime and intensive care unit crowding, while other studies have used Bertsimas–Sim budget sets or Wasserstein non-distributable methods.[2,7,37–39] Although these models are conservative, they

ensure robust decision-making in the absence of reliable probabilistic data. Also, simulation and simulation-optimization methods have been widely used for patient flow analysis and policy evaluation, especially of the discrete event type.[26]

Deep reinforcement learning, as an end-to-end, model-free framework, has the potential to overcome the limitations of traditional methods in OR scheduling.[20,40] In this approach, instead of solving MIP models daily, the DRL agent learns a policy that maps the system state (current schedule, patients in queue, and new arrivals) to scheduling decisions, which is continuously updated through simulation or real-world experience.[41,42] Recent studies have shown various applications of DRL in the healthcare domain: for example, Lee and Lee[43] reported significant reductions in waiting times and penalties compared to classical rules by training a deep Q-network for emergency patient scheduling. In the inpatient domain, Rabbani et al.[13] used an online bandit reinforcement learning algorithm to control patient admissions with random stays, outperforming static methods. A review by Dai and Shi[28] focused on a single type of problem, such as backlog resolution, triage of emergency patients, or scheduling elective surgeries. In contrast, existing models are mostly hybrid or two-stage; for example, Eshghali et al.[15] reserved capacity for emergency patients and rearranged elective scheduling in the event of unexpected arrivals, while Heydari and Soudi[25] used a two-stage stochastic model that first creates an initial schedule with free space and then modifies it in the event of emergencies in a second stage. These staged approaches differ fundamentally from a unified, real-time DRL model.

What is lacking is a single-shot DRL method that learns, for example, a dispatch policy from all system states (including current elective patients waiting, known future demand, and any emergencies that arrive) to actions (which patient to schedule next or when to idle). Such an approach would treat emergencies not as unplanned disruptions needing separate handling, but as part of the sequential decision process. In principle, a DRL agent could learn the appropriate trade-off (e.g., cancelling an elective to accommodate an emergency) from the reward function, without manually designed threshold rules. The success of DRL in other dynamic scheduling tasks suggests its feasibility, yet the literature has not yet produced an integrated DRL model for mixed elective–emergency OR scheduling. We asserted that DRL is a promising direction to fill this gap by yielding an online scheduling policy that inherently balances elective and emergency needs without a separate rescheduling phase.

3. Problem description for dynamic operating room scheduling

Traditional OR scheduling assigns a set of elective surgeries with known durations to available ORs over a fixed planning horizon (one day in this study) to minimize total overtime. In this static framework, elective cases can be planned well in advance, whereas emergency surgeries arrive unpredictably and must be treated promptly. The primary goal is to ensure that emergency cases are assigned to ORs without delay while minimizing total overtime, thereby striking an optimal balance between emergency patient care and overall schedule efficiency. The mathematical formulation for this scheduling problem is detailed in this section. In our model, we considered the following key aspects:

(i) Elective surgeries: A predetermined set I of elective surgeries, with all patients present at the start of the surgical shift and specific surgery durations.
(ii) Operating rooms: A set R of available ORs is provided. Every OR is available from the beginning of the shift, and each OR can host at most one surgery at a time.
(iii) Emergency surgeries: Emergency surgeries occur dynamically and have a higher priority than elective surgeries. If an emergency patient arrives and an OR is available, the emergency surgery must commence immediately.
(iv) Assumptions: All proposed models assume:
- Each surgery's surgeon and required resources are known in advance.
- No surgery can be interrupted once started.
- All required resources (rooms, equipment, and pre- and post-operative care) are sufficiently available, so delays occur only due to scheduling conflicts.
- The scheduling horizon is one day, with overtime incurred for surgeries that exceed the regular OR time.
- Hospital policies define priority classes (emergency, urgent, and elective) and fairness rules.
- Historical data used for training may be incomplete or biased.

- The scheduler (agent) observes the current system state with limited latency.
- Scheduling decisions must respect mandatory rest periods, maximum consecutive work hours, and other regulatory constraints.

(v) Parameters:
- I: Set of elective surgeries, indexed by i (and i' for pairwise comparisons).
- R: Set of available ORs, indexed by r.
- d_i: Duration of surgery i (in minutes), for all $i \in I$.
- Reg: The daily regular OR time (in minutes) after which overtime is incurred. The surgical day is assumed to start at time 0.
- Over: The maximum amount of overtime (in minutes) allowed for each OR.
- M: A sufficiently large positive number used for modeling purposes (big-M).

(vi) Decision variables:
- A_{ir}: A binary variable equal to 1 if surgery i is assigned to OR r; 0 otherwise, for all $i \in I$ and $r \in R$.
- $B_{ii'r}$: A binary variable equal to 1 if, for surgeries i and i' $(i \neq i')$, both are assigned to OR r and surgery i precedes surgery i' (not necessarily immediately); 0 otherwise.
- s_i: A non-negative continuous variable representing the start time of surgery i (in minutes), for all $i \in I$.
- o_r: A non-negative continuous variable representing the overtime incurred by OR r, for all $r \in R$.

(vii) Objective function: The objective is to minimize the total overtime across all ORs, which is equivalent to minimizing the overall makespan of the schedule:

$$\text{Minimize } z = \sum_{r \in R} o_r \tag{1}$$

(viii) Constraints:

$$\sum_{r} A_{ir} = 1, \forall i \tag{2}$$

$$B_{ii'r} + B_{i'ir} \leq 1, \quad \forall i, i' \in I \text{ with } i \neq i',\ \forall r \in R \tag{3}$$

$$A_{ir} + A_{i'r} \leq 1 + B_{ii'r} + B_{i'ir}, \quad \forall i, i' \in I \text{ with } i \neq i',\ \forall r \in R \tag{4}$$

$$A_{ir} - A_{i'r} \leq 1 - B_{ii'r} - B_{i'ir}, \quad \forall i, i' \in I \text{ with } i \neq i',\ \forall r \in R \tag{5}$$

$$s_i \geq s_{i'} + d_{i'} - M\left(1 - \sum_{r} B_{i'ir}\right), \quad \forall i, i' \in I \text{ with } i \neq i' \tag{6}$$

$$s_i + d_i \leq \text{Reg} + \text{Over}, \quad \forall i \in I \tag{7}$$

$$o_r \geq s_i + d_i - Reg - M(1 - A_{ir}), \quad \forall i \in I,\ \forall r \in R \tag{8}$$

Equation 2 ensures that each surgery is assigned to precisely one OR. **Equation 3** ensures that for any pair of surgeries i and i' assigned to the same OR, at most one precedence relationship can be established. **Equation 4** guarantees that if surgeries i and i' are both assigned to OR r, then a precedence relation must exist between them. **Equation 5** ensures that if a precedence relation is established between surgeries i andi', then they must be assigned to the same OR. **Equation 6** ensures that the start time of surgery i must be no earlier than the finish time of its preceding surgery i' (if such a precedence is defined). We used a big-M constraint to enforce **Equation 6**. Every surgery must be completed within the allowable time of the day (regular time plus maximum overtime). The overtime for an OR r is defined as the excess time beyond the regular time when the last assigned surgery finishes, as shown in **Equations 7 and 8**.

In **Section 4**, we introduce a DRL framework that handles emergency arrivals in real time, eliminating the need for a separate rescheduling phase.

4. Methodology

In this work, we proposed a DRL approach to DORS. In our setting, all elective surgeries are available at the start of the day for a set duration, with their start times to be determined. The primary goal was to ensure that emergency cases, which have a higher priority, are assigned to ORs without delay while minimizing total overtime, thereby striking an optimal balance between emergency patient care and overall schedule efficiency. If an emergency patient arrives and an OR is idle, it is immediately scheduled; if not, the emergency patient waits until an OR becomes available. Unlike models that incorporate a separate "rescheduling" phase, our method continuously updates the schedule in real time through sequential decision-making. At every decision moment dur-

ing the day, the algorithm evaluates the current state and directly outputs which patient should enter which OR. As illustrated in **Figure 1**, our overall DRL framework consists of real-time state updates, a multi-layer perceptron (MLP)-based state embedding, and actor–critic networks for policy learning.

4.1. Markov decision process formulation for dynamic operating room scheduling

We presented the DORS problem as an MDP, defined by the tuple (S, AS, P, R, γ):

(i) State space (S): At each decision epoch t, the state s_t encapsulates:
- Step 1. Assignment matrix (A): A binary matrix of dimensions $|I|\times|R|$, where $\mathrm{A}_{ir} = 1$ indicates that elective surgery i is assigned to OR r; otherwise, $\mathrm{A}_{ir} = 0$.
- Step 2. Start times($\{si\}$): A vector of non-negative continuous variables representing the scheduled start times for each surgery $i \in I$.
- Step 3. Surgery durations($\{di\}$): A vector of specific durations for each surgery $i \in I$.
- Step 4. Overtime ($\{Or\}$): A vector of non-negative continuous variables where O_r denotes the overtime incurred by OR $r \in R$, calculated based on the finishing time of the last surgery relative to the regular operating hours.
- Step 5. Emergency surgery status(E_t): A set indicating the current status of emergency surgeries at time t (e.g., binary flags representing the presence of pending emergency surgeries).

These comprehensive state representations provide the DRL agent with all necessary information regarding current surgery assignments, scheduling, and emergency events.

(ii) Action space (AS): An action a_t at time t involves scheduling a surgery to an OR at a specific start time, represented as a triplet:

$$a_t = (i, r, s_i) \tag{9}$$

where i is the index of the surgery to be scheduled (either elective or emergency), r is the index of the OR, and s_i is the proposed start time for surgery i.
- Validity conditions: An action a_t is valid if:
- Surgery i has not been previously scheduled.
- OR r is available at time s_i (i.e., no scheduling conflicts).
- The proposed start time adheres to operational constraints (e.g., the surgery's end time $s_i + d_i$ does not exceed allowable working hours).

Invalid actions, such as those causing scheduling conflicts, are filtered out during decision-making using action-masking mechanisms.

(iv) Transition dynamics (P): Upon executing action a_t :
- The elective surgery is scheduled in the designated OR with the specified start time.
- The finish time is computed as $s_i + d_i$, and the assignment matrix A is updated accordingly.

Overtime o_r for OR r is recalculated as:

$$o_r = \max\{(s_i + d_i) - Reg, 0\} \tag{10}$$

where Reg denotes the daily regular OR time (in minutes). If an emergency surgery arrives, the state updates immediately: if an OR is idle, the emergency surgery is scheduled promptly; otherwise, it joins a queue awaiting availability.

(v) Reward function (R): The reward function is designed to minimize total overtime. At each decision epoch t, the reward is defined as:

$$\begin{aligned} r_r = -(&\text{Total overtime}\,(s_{t+1}) \\ &-\text{Total overtime}(s_t)) \end{aligned} \tag{11}$$

with:

$$\text{Total overtime}(s_t) = \sum_{r\in R} o_r \tag{12}$$

Thus, actions that reduce total overtime yield positive rewards, while those that increase it result in negative rewards.

(vi) Discount factor (γ): We set the discount factor γ to 1 to reflect the episodic nature of the scheduling problem within a single day, ensuring that future rewards (i.e., reductions in overtime) are valued equally with immediate rewards.

4.2. Network architecture

Our DRL framework employed an actor–critic architecture and was trained via PPO.

4.2.1. State embedding network

An MLP is used to convert the raw state s_t into a compact representation. The input comprises the flattened features from:

- the assignment matrix A,
- current start times,
- specific surgery durations,

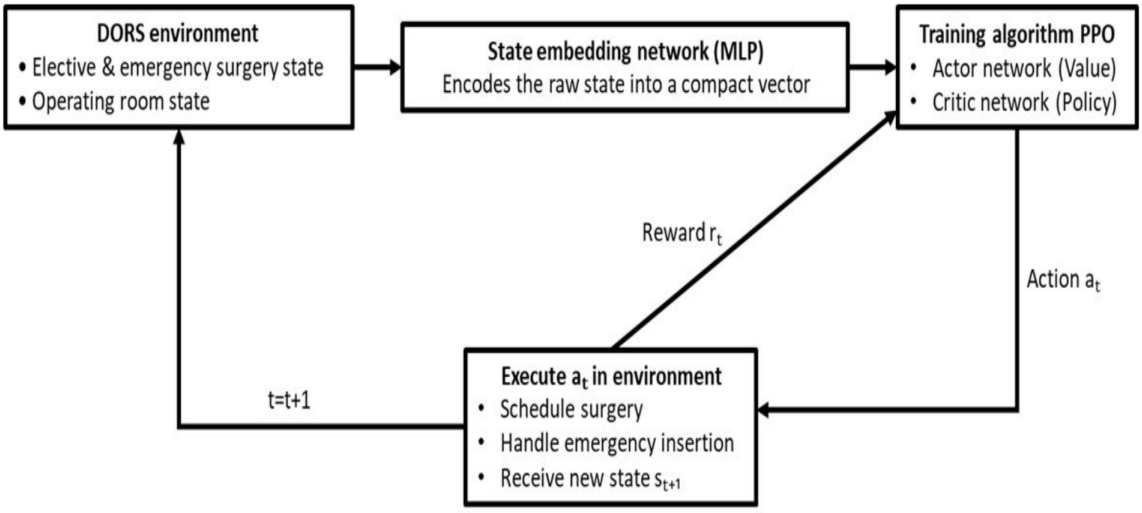

Figure 1. Overview of the deep reinforcement learning framework for dynamic operating room scheduling (DORS). Abbreviation: MLP: Multi-layer perceptron; PPO: Proximal policy optimization.

- overtime for each OR, and
- emergency surgery status.

The MLP consists of an input layer, hidden layers (e.g., with 128 and 64 neurons, activated by the Tanh function), and an output layer that generates the state embedding vector.

4.2.2. Actor network

Based on the state embedding, the actor network produces a probability distribution over the valid actions, i.e., the available surgery–OR pairs with feasible start times. It typically includes:

- One or two fully connected layers (e.g., with 64 neurons each using Tanh activation), and
- A final softmax layer that outputs the probabilities, ensuring invalid actions (determined via action-masking) receive zero probability.

4.2.3. Critic network

The critic network estimates the value function $V(s_t)$, representing the expected cumulative overtime reduction from state s_t. Its architecture is similar to that of the actor network, but it ends with a single scalar output.

4.3. Training procedure

Our DRL agent was trained over multiple episodes, each corresponding to one complete "day" of OR activity. Unlike traditional two-stage approaches that require a separate rescheduling phase when emergencies arrive, our method continuously updates the schedule in real time through sequential decision-making. At each decision step, the agent schedules exactly one surgery—either elective or emergency—based on the current state, immediately updates the environment, and receives a reward reflecting the change in total overtime. The start steps are as follows:

(i) Step 1: Episode initialization. At the start of each episode n, we initialize the environment state s_0 as follows:
- Assignment matrix A is set to all zeros (no surgery assigned).
- Overtime vector $o_r = 0$ for all ORs r.
- Elective surgery durations and the emergency arrival schedule are loaded.
- Emergency queue is empty.
- Within-episode rollout.

We then iterate decision steps $t = 0, 1, \ldots$ until all surgeries are scheduled and the episode terminates.

(ii) Step 2: Observe state s_t. The state includes the current assignment matrix A, the start times of scheduled surgeries, the remaining elective durations, the current overtime per OR, and any pending emergency flags.

(iii) Step 3: Compute state embedding. An MLP encodes the raw state features into a compact vector representation h_t.

(iv) Step 4: Actor–critic forward pass. Actor uses h_t to generate a probability distri-

bution $\pi_\theta(a \mid \mathrm{s_t})$ over all valid scheduling actions (room–surgery–start-time triplets), employing action masking to eliminate infeasible moves. Critic estimates the state value $V_\theta(s_t)$.

(v) Step 5: Sample and execute action a_t. We sample $a_t \sim \pi_\theta(\ . \mid s_t)$ (or take the arg max when evaluating deterministically), then apply it in the environment: assign the chosen surgery to the chosen OR at the specified start time, immediately insert any arriving emergency if an OR is idle (or enquire it otherwise), update A, and recalculate each o_r.

(vi) Step 6: Receive reward r_t. It is calculated as defined earlier.

(vii) Step 7: Store transition. We append (s_t, a_t, r_t, s_{t+1}) to the rollout buffer for subsequent optimization.

(viii) Step 8: Check termination. The episode ends when all surgeries (elective and emergency) are scheduled, or the maximum allowable horizon (including overtime) is reached.

For the finish part, the PPO algorithm builds on the ideas of trust region policy optimization but replaces complex constraints with a clipped surrogate objective that is straightforward to implement and delivers state-of-the-art performance across many reinforcement-learning benchmarks. While we have detailed the layer dimensions and activation functions for our MLP, we also employed generalized advantage estimation (GAE) to compute the advantage function $\hat{A}_t$, which has been shown to reduce variance and improve sample efficiency during PPO training. The PPO algorithm optimizes a composite objective composed of two main loss terms.

At each update step, given a batch of transitions $\left(\mathrm{s_t}, \mathrm{a_t}, \widehat{\mathrm{A}}_\mathrm{t}\right)$, PPO maximizes the following clipped surrogate objective with respect to the policy parameters θ:

$$L_t^{CLIP}(\theta) = \hat{E}_t\left[min\left(r_t(\theta)\hat{A}_t, clip\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \tag{13}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}$ is the likelihood ratio between the updated policy π_θ and the reference (old) policy$\pi_{\theta_{old}}$, A^t is an estimate of the advantage at time t, ϵ is a small hyperparameter (e.g., $\epsilon = 0.2$) that specifies the clipping range, and clip$(x, 1-\epsilon, 1+\epsilon)$ truncates x to lie in $[1-\epsilon, 1+\epsilon]$.

By taking the minimum of the unclipped and clipped objectives, PPO ensures that policy updates do not move $r_t(\theta)$ too far from 1, thereby limiting the policy shift and improving stability. In practice, we augmented the clipped surrogate objective with two additional loss terms:

- Value-function loss $L_t^V(\phi)$, which is the mean-squared error between the critic's value estimate $V_\phi(s_t)$ and the empirical return r_t.
- Entropy bonus $L_t^E(\theta)$, which encourages exploration by maximizing the policy's entropy.

The composite loss to minimize (or negative objective to maximize) is:

$$\mathcal{L}_t(\theta, \emptyset) = -c_p L_t^{CLIP}(\theta) + c_v L_t^V(\phi) - c_e L_t^E(\theta) \tag{14}$$

with coefficients $c_p > 0$ weighting the policy (clipped surrogate) term, $c_v > 0$ weighting the value-function loss, and $c_e > 0$ weighting the entropy bonus. The proposed algorithm is introduced in **Algorithm 1**.

Algorithm 1

For episode = 1 to N_ep do
For step = 1 to T_max do
While environment is not terminal do
Collect experience (s_t, a_t, r_t, s_ {t+1})
If terminal state reached then
Break
End if
End while
End for
For iteration = 1 to N_upd do
Compute advantage A_t and update actor π_θ and critic V_φ
End for
If (episode mod f_ver == 0) then
Perform performance verification of the policy model
End if
End for

The environment continuously integrates emergency surgeries: upon each emergency arrival, an idle OR immediately executes the case; otherwise, the emergency is enquired until the next OR becomes available. In summary, we cast DORS as an MDP in which elective surgeries—with specific durations and variable start times—were scheduled sequentially to minimize total overtime. The state s_t comprises the current assignment matrix A , scheduled start times, specific surgery durations, accumulated overtime per OR, and emergency-arrival indicators. At each decision step, the DRL agent selects a single action (assigning either an elective or emergency surgery), the environment

Table 1. Hyperparameter settings for the proximal policy optimization algorithm.

Hyperparameters	**Values**
Number of total episodes (N)	100,500
Performance verification every V episodes	3,20
Number of policy updates (K)	1
Learning rate (lr)	1×10^{-3}
Coefficient for policy term (c_p)	2
Coefficient for entropy term (c_e)	0.01
Coefficient for critic term (c_v)	1
Discount factor (γ)	1
Clipping coefficient (ϵ)	0.2

advances to s_{t+1}, and a reward equal to the negative increment in total overtime is issued. An actor–critic network, trained with PPO, then updates the scheduling policy to maximize long-term overtime reduction.

5. Experimental setting and results

In this section, we first describe our computational environment and the synthetic OR-scheduling benchmarks. We then introduce two state-of-the-art baselines—reactive MIP (RMIP) and the overload-slack index heuristic (OSI) by Miao and Wang[44]—before detailing our evaluation metrics. We present training-curve analyses that demonstrate rapid convergence and subsequently report extensive empirical comparisons on small, medium, and large instances.

5.1. Experimental setting

5.1.1. Hardware and software

All experiments ran on an Intel® Core™ i5-4210U CPU (1.70 GHz, Turbo Boost to 2.40 GHz), with 8 GB of RAM and an NVIDIA GeForce 840M GPU. The software stack comprised Python 3.12.7 and PyTorch 2.5.1. Our PPO implementation used the Adam optimizer with learning-rate decay, gradient clipping, and mini-batch updates, following industry best practices. Key hyperparameters are listed in **Table 1**. The parameters were empirically adjusted through several trial runs to identify the optimal combination. The goal of parameter tuning was to improve convergence, increase response quality, and avoid getting stuck in local optima.

5.1.2. Synthetic data generation

To overcome the scarcity of real-world OR scheduling data with emergency arrivals, we generated a diverse suite of test instances that reflect typical hospital operations. Test problems were generated according to previous studies.[31,45] The specifications are as follows:

- OR count (R): {2, 3, 5, 10, 15, 20, 25, 30, 35, 40}
- Regular operating time: 480 minutes
- Overtime window: 120 minutes
- Elective surgeries (I): 10–200 per instance; durations sampled uniformly from [50, 200] minutes
- Emergency arrivals (E): 1–3 per day with arrival times uniform in [0, 480] minutes
- Dataset split: 500 training, 100 validation, 150 test instances (no overlap)

This design ensures (i) scalability across small to very large departments, (ii) realism through typical procedure lengths and emergency "shock" events, and (iii) reproducibility via fixed random seeds.

5.1.3. Baseline methods

We benchmarked against two recently proposed dynamic scheduling methods pioneered by Miao and Wang:[44]

(i) Reactive MIP: A two-stage approach that first solves an ex-ante MIP for elective surgeries, then re-optimizes—including emergencies—via a second MIP. We implemented both stages in PuLP with a 10,000 s time limit, mirroring the original formulation.

(ii) Overload-slack index (OSI) heuristic: A fast, rule-based dispatcher that computes per-OR slack after elective assignment freezes the most underloaded rooms, and greedily allocates emergencies to minimize overtime. This $O(|R|+|I|+|E|)$ method runs in milliseconds.

5.1.4. Evaluation metrics

For each method m on test instance k, we measured:

- Total overtime (OT): It is calculated as defined earlier.
- Computation time (T): We measure the time $T_{m,k}$ (in seconds) required by method m to produce a complete scheduling on instance k.
- Interquartile mean (IQM): To obtain a robust summary of total overtime performance, we discard the lowest 25% and highest 25% of the observed $\{OT_{m,k}\}_{k=1}^{K}$ values, then compute the

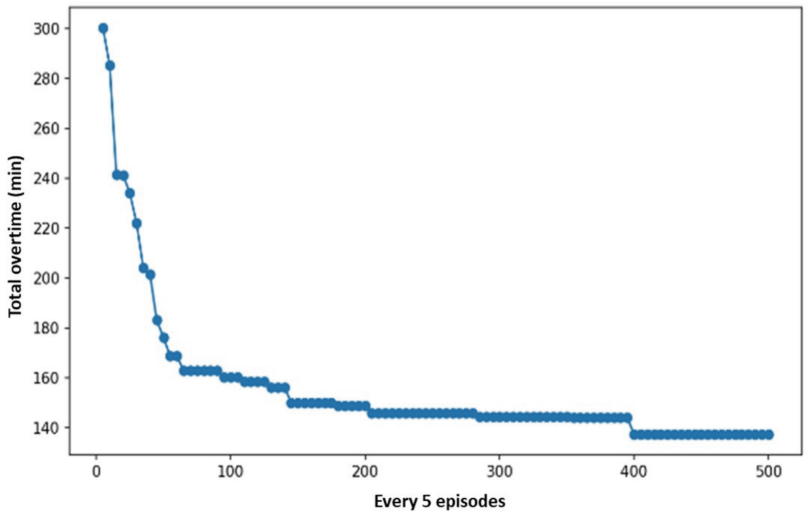

Figure 2. Training curve of the policy model trained under the uniform distribution..

arithmetic mean of the remaining middle 50%:

$$IQM_m = \frac{1}{K/2} \sum_{j=K/4+1}^{j=3K/4} OT_{m,j} \qquad (15)$$

where $OT_{m,1} \leq OT_{m,2} \leq \cdots \leq OT_{m,K}$ are the sorted overtime figures.

- Average probability of improvement (API): To quantify how often method m strictly outperforms a reference method b in terms of overtime, we define:

$$API_{m,b} = \frac{1}{K} \sum_{k=1}^{K} 1\{OT_{m,k} < OT_{b,k}\} \qquad (16)$$

where $1\{\cdot\}$ is the indicator function. An API close to 1 indicates that m yields lower overtime on almost every instance when compared to b.

5.2. Training curve analysis

We shaped the reward at each step as the negative incremental change in total overtime.

5.2.1. Uniform arrivals

To verify convergence, we periodically evaluated the policy on 10 held-out test instances (**Figure 2**). The average total overtime declined monotonically and stabilized after approximately 50'episodes, indicating fast, consistent learning under deterministic arrivals.

5.2.2. Poisson arrivals

Early-phase total overtime exhibited higher variance due to stochastic arrivals but followed a clear downward trend, eventually reaching a plateau, demonstrating robust convergence in uncertain environments (**Figure 3**).

These results confirm that our incremental overtime reward reliably guides the policy toward efficient schedules in both deterministic and stochastic settings.

5.3. Empirical evaluation across scales

To thoroughly assess our DRL performance, we randomly selected test instances at three scaling levels—small, medium, and large—covering a broad spectrum of OR counts and total cases. For each instance, we reported the mean total overtime and runtime (T) averaged over two emergency-arrival models (uniform and Poisson). This dual-distribution averaging ensures that our results are robust to different stochastic arrival patterns.

5.3.1. Small-scale instances

Small-scale instances involved a limited number of decision makers, variables, or elements, such as the number of patients, ORs, or surgeries in a scheduling problem. These samples were used for initial model validation or for comparison with exact solutions (exact methods), since exact solutions are time-consuming. For this purpose, nine instances with $R \times (I + E) \in \{3 \times 12,\ 3 \times 13,\ 5 \times 20,\ 5 \times 21,\ 10 \times 38, 10 \times 39, 10 \times 40,\ 10 \times 41,\ 10 \times 46\}$ (see **Table 2**) were sampled to represent lightly loaded departments.

In all nine small-scale instances, our DRL framework outperformed RMIP and OSI (API = 1), reducing average total overtime by approximately 120 min and being 10^2–$10^3\times$ faster than RMIP (see **Table 3**).

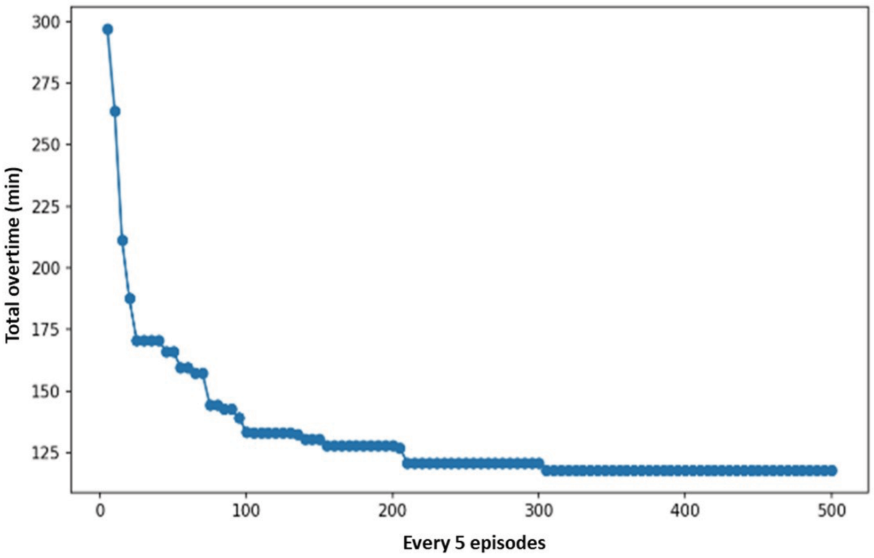

Figure 3. Training curve of the policy model trained under the Poisson distribution..

Table 2. Comparison of results across RMIP, OSI, and our proposed method in small-scale instances.

Instances	**Overtime (min)**			**Runtime (s)**		
R×(I+E)	Our proposed method	RMIP	OSI	Our proposed method	RMIP	OSI
3×12	103	223	297	0.08	12.3	0.07
3×13	338	458	517	0.10	18.0	0.09
5×20	25	145	217	0.50	24.0	0.30
5×21	177	297	327	0.58	28.0	0.42
10×38	0	73	190	0.90	35.0	0.46
10×39	144	264	420	0.92	56.0	0.49
10×40	125	245	323	0.96	82.0	0.52
10×41	140	260	367	0.99	175.0	0.69
10×46	253	373	573	1.30	264.0	1.10

Abbreviations: OSI: Overload slack index; RMIP: Reactive mixed-integer programming.

Table 3. Evaluation metrics in small-scale instances.

Methods	**IQM (overtime; min)**	**API (vs. RMIP)**	**API (vs. OSI)**
Our proposed method	146.50	1	1
RMIP	266.50	-	1
OSI	359.25	-	-

Abbreviations: API: Average probability of improvement; IQM: Interquartile mean; OSI: Overload slack index; RMIP: Reactive mixed-integer programming.

5.3.2. Medium-scale instances

Medium-scale instances involved larger sets of data (e.g., 20 to 50 surgeries or decisions in the planning horizon) that exact methods can still solve in an acceptable time or by meta-heuristics with multiple iterations. They were used for parameter tuning and algorithm stability checks. For this purpose, six instances with $R \times (I + E) \in \{15 \times 55,\ 15 \times 57,\ 20 \times 74,\ 20 \times 78,\ 25 \times 93,\ 25 \times 95\}$ (see **Table 4**) were chosen to reflect moderately loaded settings.

In medium-scale instances, our DRL framework achieved near-zero overtime in half of the surgeries (IQM = 13.5 min), outperformed RMIP and OSI in 5 of 6 instances, and completed in < 2 s, whereas RMIP took minutes (see **Table 5**).

5.3.3. Large-scale instances

Large-scale instances involved a large number of decisions, resources, or constraints, such as

Table 4. Comparison of results across RMIP, OSI, and our proposed method in medium-scale instances.

Instances	**Overtime (min)**			**Runtime (s)**		
R×(I+E)	Our proposed method	RMIP	OSI	Our proposed method	RMIP	OSI
15×55	0	113	256	1.40	352	1.30
15×57	214	334	518	1.42	412	1.41
20×74	0	0	0	1.56	523	1.53
20×78	11	156	216	1.62	687	1.58
25×93	0	21	78	1.73	712	1.86
25×95	43	306	493	1.75	993	1.91

Abbreviations: OSI: Overload slack index; RMIP: Reactive mixed-integer programming.

Table 5. Evaluation metrics in medium-scale instances.

Method	**IQM (overtime; min)**	**API (vs. RMIP)**	**API (vs. OSI)**
Our proposed method	13.50	0.83	0.83
RMIP	149.00	-	0.83
OSI	260.75	-	-

Abbreviations: API: Average probability of improvement; IQM: Interquartile mean; OSI: Overload slack index; RMIP: Reactive mixed-integer programming.

Table 6. Comparison of results across RMIP, OSI, and our proposed method in large-scale instances.

Instances	**Overtime (min)**			**Runtime (s)**		
R×(I+E)	Our proposed method	RMIP	OSI	Our proposed method	RMIP	OSI
30×112	0	0	52	1.86	1,125	1.93
30×117	158	310	493	1.83	1,401	1.89
30×124	14	162	215	2.01	1,526	2.01
35×129	0	73	177	2.14	1,896	2.11
35×142	28	172	263	2.30	2,257	2.27
35×156	0	19	75	2.90	3,682	2.86
40×170	37	132	217	3.20	4,896	3.01
40×184	0	0	38	3.18	6,087	3.24
40×198	27	107	193	3.60	8,426	3.70

Abbreviations: OSI: Overload slack index; RMIP: Reactive mixed-integer programming.

Table 7. Evaluation metrics in large-scale instances.

Method	**IQM (overtime; min)**	**API (vs. RMIP)**	**API (vs. OSI)**
Our proposed method	13.8	0.78	1.00
RMIP	98.6	-	0.78
OSI	175.4	-	-

Abbreviations: API: Average probability of improvement; IQM: Interquartile mean; OSI: Overload slack index; RMIP: Reactive mixed-integer programming.

more than 100 surgeries, multiple ORs, and multiple medical teams. These examples are usually real or industrial and are impossible or highly time-consuming to address with exact methods. Meta-heuristics were used to test efficiency and scalability. Nine instances with $R \times (I + E) \in \{30 \times 112,\ 30 \times 117,\ 30 \times 124,\ 35 \times 129,\ 35 \times 142,\ 35 \times 156,\ 40 \times 170,\ 40 \times 184,\ 40 \times 198\}$ (see **Table 6**) were sampled to challenge the scheduler under heavy load.

Even with up to 198 total surgeries, our proposed DRL framework maintained an IQM of 13.8 min, outperformed RMIP in 7 out of 9 instances (API = 0.78) and OSI in every instance (API = 1.00), and completed within 4 s (see **Table 7**).

6. Conclusion

We have presented a novel, end-to-end DRL framework for DORS that seamlessly integrates elective and emergency cases without resorting to offline master schedules or reactive rescheduling. Modeling the problem as an MDP and training an actor–critic agent with PPO and GAE enables continuous, online decision-making that directly optimizes overtime reduction. Extensive experiments on synthetic benchmarks of varying scale and stochasticity revealed that our DRL (i) dramatically lowered total overtime compared to both the RMIP approach and the heuristic dispatcher; (ii) maintained robust performance across diverse department sizes and arrival processes; and (iii) computed high-quality schedules in seconds, suitable for real-time use.

These findings underscore the promise of DRL for high-impact healthcare operation problems, where traditional optimization methods falter under uncertainty and time constraints. In the future, the proposed DRL framework can be evaluated on real-world hospital scheduling data, extending its state representation to include downstream bed and staffing constraints and exploring multi-agent architectures for coordinated scheduling across multiple operating suites.

While the proposed DRL framework demonstrates promising potential for DORS, integrating it into real-world hospital environments faces several critical barriers. Institutional and policy constraints often limit the flexibility of automated scheduling systems, as hospital management procedures, staff unions, and departmental hierarchies typically require manual oversight and approval for schedule modifications. These administrative policies may hinder the full automation of DRL-based decision-making, necessitating the development of hybrid systems that balance algorithmic recommendations with human supervision.

Moreover, data privacy and security represent significant challenges in the healthcare domain. Training DRL models requires access to sensitive patient and operational data, including personal identifiers, medical histories, and surgery details. Ensuring compliance with data protection regulations, such as the Health Insurance Portability and Accountability Act and General Data Protection Regulation, is essential to prevent unauthorized access and misuse. Techniques such as federated learning, data anonymization, and secure model deployment can mitigate these concerns. However, they also increase system complexity and computational cost. Finally, ethical considerations must be addressed before large-scale implementation. Automated scheduling algorithms must ensure fairness, transparency, and accountability in decision-making—particularly in prioritizing emergency versus elective cases. Ethical dilemmas may arise if algorithmic policies unintentionally favor certain patient groups or compromise the autonomy of medical staff. Therefore, incorporating explainable artificial intelligence mechanisms and involving clinicians in model validation and governance processes are crucial steps toward building trust and ensuring responsible adoption of DRL-based scheduling systems in hospital settings.

Future research can build upon the current study in several important directions. First, applying the proposed DRL-based scheduling framework to real-world hospital datasets would enable more realistic validation under operational constraints, such as surgeon availability, patient priorities, and emergency frequencies. Second, conducting scalability analyses across larger, more complex hospital networks could evaluate how the model performs as the number of ORs, staff members, and surgery types increases. Third, the framework could be integrated with existing hospital information systems or electronic health record platforms to enable real-time data exchange and automated decision support, ensuring practical deployment feasibility. Fourth, comparative performance benchmarking against other advanced machine learning and optimization algorithms (e.g., deep Q-networks, actor–critic variants, or meta-heuristic hybrids) would provide quantitative insights into the relative strengths and weaknesses of the proposed method. Fifth, the experimental setting was limited to a hardware configuration; scalability testing in more modern or diverse computing environments would be useful. Finally, extending the model to incorporate multi-objective optimization criteria—such as patient waiting time, resource utilization, and cost efficiency—could further enhance its clinical and operational applicability.

Acknowledgments

None.

Funding

None.

Conflict of interest

The authors declare they have no competing interests.

Author contributions

Conceptualization: Farshad Ahmadian, Mohammad Bagher Fakhrzad
Data curation: Farshad Ahmadian
Formal analysis: Farshad Ahmadian, Mohammad Bagher Fakhrzad
Methodology: Farshad Ahmadian
Software: Farshad Ahmadian
Supervision: Mohammad Bagher Fakhrzad
Validation: Mohammad Bagher Fakhrzad, Hasan Hosseini-Nasab
Writing – original draft: Farshad Ahmadian
Writing – review & editing: Mohammad Bagher Fakhrzad, Davood Shishebori

Availability of data

The synthetic datasets generated and analyzed during the current study are available from the corresponding author upon reasonable request.

AI tools statement

All authors confirm that no AI tools were used in the preparation of this manuscript.

References

1. Zhu S, Fan W, Yang S, Pei J, Pardalos PM. Operating room planning and surgical case scheduling: a review of literature. *J Comb Optim.* 2019;37(3):757-805. https://www.doi.org/10.1007/s10878-018-0322-6.
2. Addis B, Carello G, Tanfani E. Evaluating the impact of the level of robustness in operating room scheduling problems. *Healthcare.* 2024;12(20):2023. https://www.doi.org/10.3390/healthcare12202023.
3. Abdalkareem ZA, Amir A, Al-Betar MA, Ekhan P, Hammouri AI. Healthcare scheduling in optimization context: a review. *Health Technol (Berl).* 2021;11(3):445-469. https://www.doi.org/10.1007/s12553-021-00547-5.
4. Kroer LR, Foverskov K, Vilhelmsen C, Hansen AS, Larsen J. Planning and scheduling operating rooms for elective and emergency surgeries with uncertain duration. *Oper Res Health Care.* 2018;19:107-119. https://www.doi.org/10.1016/j.orhc.2018.03.006.
5. Schouten AM, Flipse SM, van Nieuwenhuizen KE, Jansen FW, van der Eijk AC, van den Dobbelsteen JJ. Operating room performance optimization metrics: a systematic review. *J Med Syst.* 2023;47(1):19. https://www.doi.org/10.1007/s10916-023-01912-9.
6. Ahmadi-Javid A, Jalali Z, Klassen KJ. Outpatient appointment systems in healthcare: a review of optimization studies. *Eur J Oper Res.* 2017;258(1):3-34. https://www.doi.org/10.1016/j.ejor.2016.06.064.
7. Shehadeh KS, Padman R. A distributionally robust optimization approach for stochastic elective surgery scheduling with limited intensive care unit capacity. *Eur J Oper Res.* 2021;290(3):901-913. https://www.doi.org/10.1016/j.ejor.2020.09.001.
8. Bovim TR, Christiansen M, Gullhav AN, Range TM, Hellemo L. Stochastic master surgery scheduling. *Eur J Oper Res.* 2020;285(2):695-711. https://www.doi.org/10.1016/j.ejor.2020.02.001.
9. Kamran MA, Karimi B, Dellaert N. A column-generation-heuristic-based Benders decomposition for adaptive allocation scheduling of patients in operating rooms. *Comput Ind Eng.* 2020;148:106698. https://www.doi.org/10.1016/j.cie.2020.106698.
10. Akbarzadeh B, Moslehi G, Reisi-Nafchi M, Maenhout B. Re-planning and scheduling of surgical cases after block release time with resource rescheduling. *Eur J Oper Res.* 2019;278(2):596-614. https://www.doi.org/10.1016/j.ejor.2019.04.037.
11. Addis B, Carello G, Grosso A, Tanfani E. Operating room scheduling and rescheduling: a rolling horizon approach. *Flex Serv Manuf J.* 2016;28(1):206-232. https://www.doi.org/10.1007/s10696-015-9213-7.
12. Gauthier JB, Legrain A. Operating room management under uncertainty. *Constraints.* 2016;21(4):577-596. https://www.doi.org/10.1007/s10601-015-9236-4.
13. Rabbani M, Zhalechian M, Farshbaf-Geranmayeh A. A robust possibilistic programming approach to multiperiod hospital evacuation planning. *Int Trans Oper Res.* 2018;25(1):157-189. https://www.doi.org/10.1111/itor.12331.
14. Cardoen B, Demeulemeester E, Belien J. Operating room planning and scheduling: a literature review. *Eur J Oper Res.* 2010;201(3):921-932. https://www.doi.org/10.1016/j.ejor.2009.04.011.
15. Eshghali M, Kannan D, Salmanzadeh-Meydani N, Esmaieeli Sikaroudi AM. Machine learning based integrated scheduling and rescheduling for elective and emergency patients. *Ann Oper Res.* 2024;332(1):989-1012. https://www.doi.org/10.1007/s10479-023-05168-x.
16. Khan J, Khan MA, Sarwar S. Improvements of Hermite-Hadamard-Mercer inequality using k-fractional integral. *Int J Optim Control Theor Appl.* 2025;15(1):155-165. https://www.doi.org/10.36922/ijocta.1568.
17. Fallahpour Y, Rafiee M, Elomri A, Kayvanfar V, El Omri A. Multi-objective planning and schedul-

ing model for elective and emergency cases under uncertainty. *Decis Anal J.* 2024;11:100475. https://www.doi.org/10.1016/j.dajour.2024.100475.
18. Miao H, Wang JJ. Scheduling elective and emergency surgeries with emergency uncertainty and waiting time limit. *Comput Ind Eng.* 2021;160:107551. https://www.doi.org/10.1016/j.cie.2021.107551.
19. Wang Y, Zhang Y, Tang J. Wasserstein distributionally robust surgery scheduling with elective and emergency patients. *Eur J Oper Res.* 2024;314(2):509-522. https://www.doi.org/10.1016/j.ejor.2023.10.026.
20. Zhao L, Zhu H, Zhang M, Tang J, Wang Y. Large-scale dynamic surgical scheduling under uncertainty by hierarchical reinforcement learning. *Int J Prod Res.* 2024;62:1-32. https://www.doi.org/10.1080/00207543.2024.2361449.
21. Ahmed A, He L, Chou CA, Firouz M, Hamasha M. Prediction-optimization approach to surgery prioritization. *J Ind Prod Eng.* 2021;39:1-15. https://www.doi.org/10.1080/21681015.2021.2017362.
22. Arab Momeni M, Mostofi A, Jain V, Soni G. COVID-19 epidemic outbreak: operating room scheduling and emergency assignment using robust optimization. *Ann Oper Res.* 2022. https://www.doi.org/10.1007/s10479-022-04667-7.
23. Amin MA, Baldacci R, Kayvanfar V. A comprehensive review on operating room scheduling and optimization. *Operational Research.* 2025;25(3). https://www.doi.org/10.1007/s12351-024-00884-z.
24. Majthoub Almoghrabi M, Sagnol G. Surgery scheduling in flexible operating rooms by using a convex surrogate model of second-stage costs. *Eur J Oper Res.* 2025;321(1):23-40. https://www.doi.org/10.1016/j.ejor.2024.07.036.
25. Heydari M, Soudi A. Predictive/reactive planning and scheduling of a surgical suite with emergency patient arrival. *J Med Syst.* 2016;40(1):30. https://www.doi.org/10.1007/s10916-015-0385-1.
26. Lopes J Sr, Guimarães T, Duarte J, Santos M. Enhancing surgery scheduling with artificial intelligence: metaheuristic optimisation models. *JMIR Med Inform.* 2024;13:1-11. https://www.doi.org/10.2196/57231.
27. Lin YK, Yen CH. Genetic algorithm for solving the no-wait three-stage surgery scheduling problem. *Healthcare.* 2023;11(5):739. https://www.doi.org/10.3390/healthcare11050739.
28. Dai JG, Shi P. Recent modeling and analytical advances in hospital inpatient flow management. *Prod Oper Manag.* 2021;30(6):1838-1862. doi: 10.1111/poms.13132.
29. Zhu T, Xie J, Sim M. Joint estimation and robustness optimization. *Manage Sci.* 2022;68(3):1659-1677. https://www.doi.org/10.1287/mnsc.2020.3898.
30. Lee S, Yih Y. Analysis of an open access scheduling system in outpatient clinics. *Simulation.* 2010;86:503-518. https://www.doi.org/10.1177/0037549709358295.
31. Leeftink G, Hans E. Case mix classification and benchmark set for surgery scheduling. *J Sched.* 2018;21:1-14. https://www.doi.org/10.1007/s10951-017-0539-8.
32. Britt J, Baki MF, Azab A, Chaouch A, Li X. Stochastic hierarchical approach for the master surgical scheduling problem. *Comput Ind Eng.* 2021;158:107385. https://www.doi.org/10.1016/j.cie.2021.107385.
33. Oliveira M, Bélanger V, Marques I, Ruiz A. Assessing the impact of patient prioritization on operating room schedules. *Oper Res Health Care.* 2020;24:100232. https://www.doi.org/10.1016/j.orhc.2019.100232.
34. Vali-Siar MM, Gholami S, Ramezanian R. Multi-period and multi-resource operating room scheduling under uncertainty. *Comput Ind Eng.* 2018;126:549-568. https://www.doi.org/10.1016/j.cie.2018.10.014.
35. Khalfalli M, Verny J. Fuzzy logic for stochastic operating theater optimization: a review. In: *Proc Int Conf Health Care Syst Eng.* 2020:259-270. https://www.doi.org/10.1007/978-3-030-34702-4_16.
36. Tsang MY, Shehadeh KS, Curtis FE, Hochman BR, Brentjens TE. Stochastic optimization approaches for operating room and anesthesiologist scheduling. *Oper Res.* 2024. https://www.doi.org/10.1287/opre.2022.0258.
37. Rachuba S, Werners B. Robust approach for scheduling in hospitals using multiple objectives. *J Oper Res Soc.* 2014;65:546-556. https://www.doi.org/10.1057/jors.2013.112.
38. Addis B, Carello G, Tànfani E. A Robust Optimization Approach for the Operating Room Planning Problem with Uncertain Surgery Duration. In: Springer Proceedings in Mathematics & Statistics. Springer International Publishing; 2013:175-189. https://www.doi.org/10.1007/978-3-319-01848-5_14.
39. Wang Y, Zhang Y, Tang J. Distributionally robust optimization for surgery block allocation. *Eur J Oper Res.* 2019;273(2):740-753. https://www.doi.org/10.1016/j.ejor.2018.08.037.
40. He L, Chalil Madathil S, Oberoi A, Servis G, Khasawneh MT. A systematic review of research design and modeling techniques in inpatient bed management. *Comput Ind Eng.* 2019;127:451-466. https://www.doi.org/10.1016/j.cie.2018.10.033.
41. Wu R, Zheng J, Yin X. Dynamic scheduling for multi-objective flexible job shops with machine breakdown by deep reinforcement learning. *Processes.* 2025;13(4):1246. https://www.doi.org/10.3390/pr13041246.

42. Yuan E, Wang L, Song S, Cheng S, Fan W. Dynamic scheduling for multi-objective flexible job shop via deep reinforcement learning. *Appl Soft Comput.* 2025;171:112787. https://www.doi.org/10.1016/j.asoc.2025.112787.
43. Lee S, Lee YH. Improving emergency department efficiency by patient scheduling using deep reinforcement learning. *Healthcare.* 2020;8(2):77. https://www.doi.org/10.3390/healthcare8020077.
44. Miao H, Wang JJ. Distributed surgical scheduling across collaborating hospitals. *Comput Ind Eng.* 2023;183:109462. https://www.doi.org/10.1016/j.cie.2023.109462.
45. Hooshmand F, MirHassani SA, Akhavein A. Adapting GA for operating room scheduling with endogenous uncertainty. *0*Oper Res Health Care. 2018;19:26-43. https://www.doi.org/10.1016/j.orhc.2018.02.002.

Farshad Ahmadian *is currently a Ph.D. candidate in Industrial Engineering at Yazd University, Iran. His research interests include operations research, optimization, dynamic scheduling, and the application of deep reinforcement learning in healthcare systems. He has conducted research on operating room scheduling and the integration of real-time uncertainty into decision-making models. He aims to develop advanced intelligent scheduling frameworks that enhance efficiency and resource utilization in complex systems.*
https://orcid.org/0009-0005-8859-1881

Mohammad Bagher Fakhrzad *is a Professor in the Department of Industrial Engineering at Yazd University, where he is affiliated with the Systems Optimization division. His research program focuses on deterministic, fuzzy, probabilistic, and robust multi-objective decision-making and optimization. He has extensive experience in system planning and improvement across various domains, including manufacturing, finance, energy, healthcare, inventory management, and project planning. His work also covers facility location, supply chain management, distribution network design, allocation, routing, scheduling, and transportation systems. In addition, he conducts research in data-driven modeling, including neural networks, data mining, expert systems, performance evaluation, and productivity improvement.*
https://orcid.org/0000-0001-8918-8588

Hasan Hosseini-Nasab *is a Professor in the Department of Industrial Engineering at Yazd University, affiliated with the Systems Optimization division. His research interests include the design of manufacturing and industrial systems, long-term and short-term planning, and dynamic systems. He also works extensively in the fields of renewable energy, quantitative and qualitative management, marketing project analysis, and statistical sampling. His academic contributions span scheduling, supply chain management, facility location, routing, optimization, expert systems, and advanced manufacturing systems. In addition, he conducts research on a wide range of solution methodologies, including heuristic, metaheuristic, exact, probabilistic, fuzzy, multi-objective, and multi-criteria approaches.*
https://orcid.org/0000-0002-0921-8618

Davood Shishebori *is an Associate Professor in the Department of Industrial Engineering at Yazd University, working within the Quality and Productivity division. His research interests include sustainable and resilient supply chain management, reliability metrics with a focus on robust optimization, and process improvement in quality engineering. He also works on the optimization of quality indicators and the application of data envelopment analysis (DEA) in productivity assessment. His academic contributions span quality management, system robustness, performance evaluation, and advanced optimization methods within industrial engineering.*
https://orcid.org/0000-0001-7737-6565