Towards Tailoring Reinforcement Learning to Solve the Online Surgery-Planning and Scheduling Problem

Khouloud Bennour[1], Imen Ghazouani[1], Asma Ouled Bedhief[1], Safa.Bhar Layeb[1, 2*] and Najla Omrane Aissaoui[1]

[1] LR-OASIS, National Engineering School of Tunis, University of Tunis El Manar, Tunis, Tunisia
[2] Centre Génie Industriel, Université Toulouse, IMT Mines Albi, Albi, France
[*] safa.layeb@enit.utm.tn

Abstract. This paper addresses the online surgery planning and scheduling problem for operating rooms and recovery beds. We aim to minimize the makespan by dynamically assigning surgery dates, operating rooms, and recovery beds. Our integrated framework uses a Mixed-Integer Linear Program (MILP) solved with Python's PuLP package for initial scheduling and Reinforcement Learning (RL) for real-time adjustments. The MILP provides a static schedule, while RL handles disruptions and updates schedules using experience replay and target networks for stable training. Preliminary results demonstrate that this approach effectively manages scheduling complexities, improving operational efficiency and minimizing idle times. This highlights the potential of combining MILP and RL for adaptive surgical scheduling

Keywords: Operating room, recovery bed, planning, scheduling, planning horizon, surgery, reinforcement learning, makespan.

1 Introduction

The healthcare sector plays a vital role worldwide, delivering essential medical services to diverse populations. Continuous advancements in medical technology have significantly expanded the capacity to treat and heal patients through surgical procedures. This progress has increased the need for effective management of critical resources, such as operating rooms, medical equipment, surgical teams, anesthesia, and recovery beds. As the demand for surgical procedures grows, these resources, especially operating rooms and recovery beds, have become key bottlenecks in hospitals. Operating rooms alone account for 40-60% of total hospital expenses [1] Thus, optimizing their use is crucial for maintaining hospital efficiency and ensuring high-quality patient care.

Integrating the planning and scheduling of operating rooms and recovery beds is increasingly recognized as essential. Operating room planning involves setting surgery dates for patients, while scheduling focuses on assigning patients to specific rooms and sequencing their surgeries throughout the day. Including recovery beds in this integration ensures seamless post-operative care, preventing delays and optimizing resource

S. Krichen et al. (eds.), *Proceedings of International Conference on Decision Aid and Artificial Intelligence (ICODAI 2024)*, Atlantis Highlights in Intelligent Systems 12,
https://doi.org/10.2991/978-94-6463-654-3_14

use. This integrated approach aims to enhance coordination between these tasks, improving the overall performance and efficiency of the surgical unit. Our research focuses on online surgery planning and scheduling for both operating rooms and recovery beds, emphasizing dynamic, real-time management to adapt to the ever-changing demands of the healthcare environment.

2 Related Work

In recent literature, various approaches have been taken to address different aspects of this problem. Some studies have focused solely on operating rooms, while others have included additional resources like recovery beds. Additionally, some research has concentrated only on planning, while others have focused on scheduling. Few studies have tackled both planning and scheduling in an online context, highlighting a gap in comprehensive solutions. For instance, Xu et al. [2] developed a stochastic control process to manage the backlog of elective surgeries by dynamically scheduling based on new arrivals, waiting times, and clinical urgency. Tsai et al. [3] introduced a stochastic optimization model for surgical scheduling, specifically focusing on a single operating room. Miao et al. [4] proposed a Mixed-Integer Linear Programming (MILP) model for preventive-reactive scheduling to handle emergency arrivals efficiently. Breuer et al. [5] employed a robust optimization framework to integrate staffing and scheduling decisions, and Allen et al. [6] analyzed the impact of schedule adjustments on patient safety, satisfaction, and operational costs.

Studies that incorporate more than just operating rooms, such as Eshghali et al. [7], developed an integrated model for OR efficiency, employing a three-phase model including emergency patient forecasts. Varmazyar et al. [8] introduced a stochastic optimization approach integrating Post-Anesthesia Care Unit (PACU) dynamics, while Saadouli et al. [9] focused on optimizing both operating rooms and recovery beds in an orthopedic surgery department in Tunisia.

In the context of surgery planning, Doulabi et al. [10] presented a two-stage stochastic model aiming to minimize costs and improve computational efficiency. Kamran et al. [11] tackled the surgery planning challenge under uncertainty, and Lamiri et al. [12] provided insights into stochastic models for operating room planning. For surgery scheduling, Yu et al. [13] addressed multi objective dynamic scheduling, Silva et al. [14] applied approximate dynamic programming, and Spangenberg et al. [15] introduced a real-time decision support system. Kamran et al. [16] also worked on the advance scheduling problem incorporating stochastic durations, and Beaulieu et al. [17] focused on a comprehensive approach to OR scheduling under uncertainty. Several studies have explored the use of reinforcement learning (RL) for surgical planning and scheduling, demonstrating its potential in handling dynamic and complex environments. For example, Xu et al. [18] utilized a piecewise decaying ε-greedy RL algorithm to dynamically schedule surgeries based on new arrivals, waiting times, and clinical urgency, effectively managing surgery backlogs during public health crises. Eshghali

et al. [19] integrated RL with machine learning forecasts in their model for online rescheduling of surgical procedures. These studies highlight the adaptability and efficiency of RL in responding to unpredictable healthcare demands, showcasing its flexibility in real-world applications. Our research builds on these advancements by focusing on an integrated RL approach for both operating rooms and recovery beds, providing a comprehensive and dynamic solution to enhance hospital efficiency and patient care.

3 Project Context

In this study, we focus on the problem of online integrated surgery planning and scheduling for operating rooms and recovery beds. Given a set of elective surgeries (patients) to be performed and a number of available operating rooms, our objectives are to minimize the makespan by dynamically assigning surgery dates, operating rooms, and recovery beds (1), Continuously adjust and optimize the schedule in real-time to handle unforeseen delays in surgery durations (2).

We propose an integrated framework (3) that uses a Mixed-Integer Linear Program (MILP) for offline planning as an initial input to provide a baseline schedule. This initial schedule determines surgery dates and resource allocations. For the online component, we employ Reinforcement Learning (RL) to handle real-time scheduling adjustments. The RL model, utilizing experience replay and target networks, allows for dynamic rescheduling to adapt to delays, ensuring optimal resource utilization and minimizing the overall makespan.

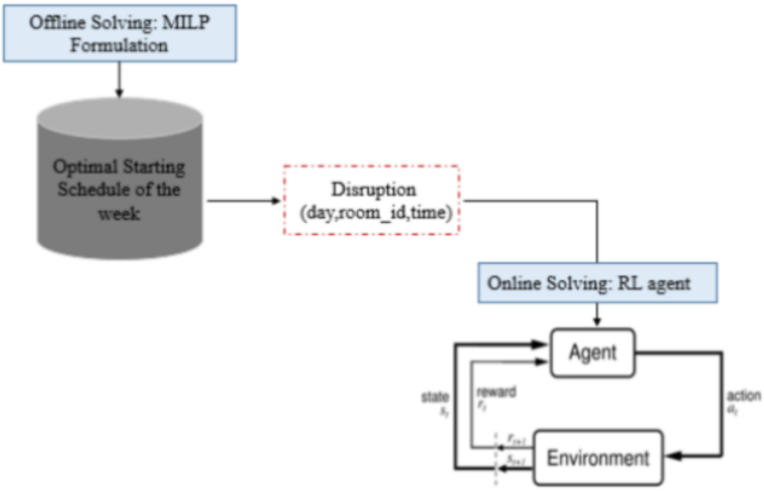

Fig.1. Integrated Framework for Surgical Planning and Scheduling

By combining the static optimality of MILP with the dynamic adaptability of RL, our approach effectively manages the complexities of surgical scheduling, ensuring efficient and high-quality patient care in a constantly changing healthcare environment. Our integrated scheduling model operates under several core assumptions, each of which shapes its generalizability and practical application.

In offline scheduling

- Operating rooms are assumed to be uniform and follow a consistent operational schedule ($Tstart$ to $Tfinish$).
- Human and instrumental resources are readily available, which simplifies planning but overlooks real-time constraints.
- Surgical durations are inclusive, covering preparatory activities and the actual procedure, which enhances precision but may limit adaptability to patient-specific variations.
- An open scheduling strategy is adopted, allowing any surgery to be performed in any available room or bed. This adds flexibility but may not apply in hospitals with specialized room allocations.

In online scheduling

- Delays are managed exclusively within operating rooms, with surgeries dynamically rescheduled within the same day to adhere as closely as possible to the original schedule.
- For overrun surgeries, subsequent operations are reprioritized, focusing on patients who have completed preoperative protocols, ensuring that critical cases progress without unnecessary delays.
- An extra hour is reserved daily to accommodate overruns, allowing flexibility to prevent cancellations. If necessary, surgeries can extend beyond regular hours to maintain high standards of patient care, though this approach may not be feasible in all settings.

These assumptions streamline scheduling and improve model stability, but they may limit adaptability in more complex, resource-constrained, or specialized hospital environments.

The rest of the paper is structured as follows: In Section 2, we provide a detailed overview of the reinforcement learning implementation. Section 3 reports some preliminary results to evaluate the performance. Finally, a conclusion section completes the paper.

4 Reinforcement Learning

Reinforcement Learning (RL) involves an agent learning to make decisions by interacting with an environment. The agent selects actions based on the current state (st),

receives rewards (rt), and observes new states, iteratively learning to optimize its actions to achieve the best outcomes.

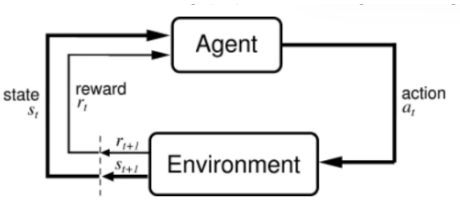

Fig.2. Agent-environment interaction loop

RL algorithms are categorized into model-based and model-free methods. Model-based algorithms use an environment model to predict future states and rewards, allowing for planning and optimization of action sequences. Examples include Monte Carlo Tree Search (MCTS). These algorithms are powerful but challenging to implement due to the difficulty in accurately modeling complex environments. On the other hand, model-free algorithms learn directly from interactions with the environment without needing a model. They are divided into on-policy methods, such as SARSA, which learn the value of the current policy by evaluating actions taken, and off-policy methods, such as Q-Learning, which learn the value of the optimal policy using data from various policies, providing greater flexibility and efficiency.

In our study, Q-Learning is chosen due to its simplicity, interpretability, and effectiveness for scheduling applications, such as operating room and recovery bed allocations. This off-policy RL algorithm provides an efficient way to learn optimal policies by estimating the value of action-state pairs through the Bellman equation, iteratively updating these values without the need for extensive computational resources. While advanced models like Deep Q-Learning (DQN) and Proximal Policy Optimization (PPO) offer additional advantages—DQN's capacity for high-dimensional state spaces and PPO's stability in policy updates—these methods also involve complex architectures and extensive tuning.

Our scheduling environment, though dynamic, does not require the depth of function approximation in DQN or the policy stability features of PPO, which are typically beneficial for continuous action spaces or environments with more fluid policy changes. Instead, Q-Learning's epsilon-greedy policy effectively balances exploration (testing new actions) with exploitation (choosing actions with the highest rewards), making it well-suited for structured, discrete scheduling tasks. Moving forward, a comparison between Q-Learning, DQN, and PPO could reveal trade-offs between computational efficiency and model robustness, particularly in higher-dimensional or more variable scheduling environments.

The Q-Learning algorithm involves the following steps:

1- Initialize Q-values: Q (s, a) for each state-action pair.
2- For each episode:
   - Observe the initial state s_1

- Loop until the terminal state or a set number of steps:
- Choose an action at using an epsilon-greedy policy.
- Perform the action, transitioning to a new state s_{t+1}
- Measure the reward r_t
- Update the Q-value based on the reward and maximum future Q-value using the Bellman equation:

$$Q^{\pi}(s_t\ , a_t\ ) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+2}\ \ |s_t\ ,\ a_t\ ] \qquad (1)$$

where Q-value which is the expected cumulative reward for taking action at in state st under policy ϵ . The RL key elements are:

- Rt: Immediate reward after action A_t
- γ: Discount factor, controlling the weight of future rewards.
- Future terms $\gamma\ Rt$+1, $\gamma\ Rt$+2: Discounted future rewards.

4.1 Exploration vs. Exploitation

A critical aspect of Q-Learning is balancing exploration and exploitation. Exploration involves trying new actions to discover their effects and potential rewards, which can lead to discovering better strategies. Exploitation, on the other hand, uses the current knowledge to select the best-known actions to maximize immediate rewards. The ϵ-greedy policy is commonly used to achieve this balance, where ϵ (ranging from 0 to 1) represents the probability of choosing a random action (exploration), and (1 - ϵ) represents the probability of choosing the best-known action (exploitation). Initially, a higher ϵ value encourages exploration, but over time, the exploration rate decays, promoting more exploitation as the agent learns.

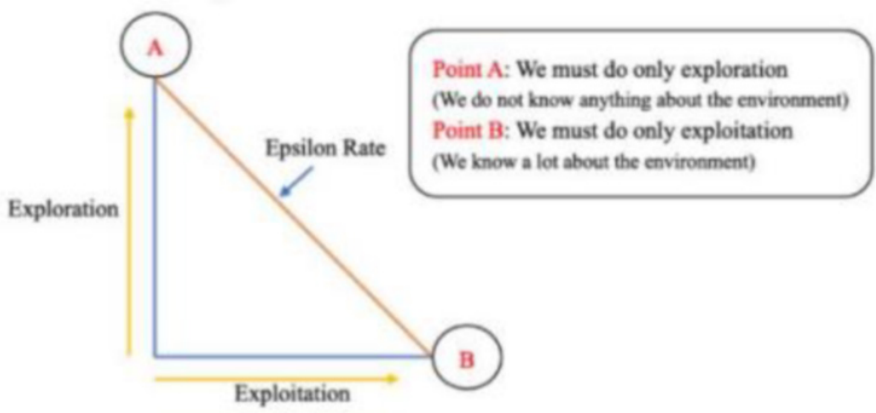

Fig. 3. The balance of epsilon rate

Reinforcement learning (RL) provides a powerful architecture for tackling complex scheduling problems in healthcare, particularly in managing operating room (OR)

schedules. This section details the methodology of applying RL to solve the online surgery planning and scheduling issues, using a tailored RL approach inspired by practical implementations.

4.2 State Representation

The state space (St) includes variables such as surgery start times, assigned rooms, surgery durations, recovery bed assignments, and the day of the week. This detailed state representation allows the agent to evaluate the current schedule comprehensively.

$$S_t = \{Start\ Time_i, Room_i, Duration_i, RB_i, Bed\ Duration_i, Day_i\}_N^{i=1} \quad (2)$$

where N is the total number of surgeries.

4.3 Action Space (A_t)

The action space consists of four actions:

$$A_t = \{a_0, a_1, a_2, a_3\} \quad (3)$$

- Action 0: Reassign rooms and beds: Reallocate the surgery to available rooms and recovery beds if there are scheduling overlaps.

$$Assign\ Room = Available\ Room\ at\ Time_t \quad (4)$$

$$Assign\ Bed = Available\ Bed\ at\ Time_t \quad (5)$$

- Action 1: Reschedule surgeries within the same day: Adjust the start time of the surgeries following the disruption to avoid overlaps within the same day.

$$Reschedule_t = Start\ Time_i + \delta_t \quad (6)$$

- Action 2: Leave the schedule unchanged: Maintain the current schedule.

$$Leave\ Unchanged_t = Current\ Schedule \quad (7)$$

- Action 3: Postpone surgeries to the next available day: Postpone the surgeries to the next available day if the current day's schedule cannot accommodate it.

$$Postpone_t = Next\ Available\ Day \quad (8)$$

$$Assign\ Room = Available\ Room\ on\ Next\ Day \quad (9)$$

$$Assign\ Bed = Available\ Bed\ on\ Next\ Day \quad (10)$$

Rewards (Rt) are designed to encourage efficient scheduling. The reward function penalizes overlaps in room and bed assignments and deviations from the initial schedule while promoting shorter makespan. An overlap occurs when multiple surgeries are scheduled simultaneously in the same room or recovery bed, leading to resource conflicts.

4.4 Reward Calculation (R_t)

The Reward is calculated as

R_t= Average Makespan+ Total Deviation + Overlap Penalty (11)

Where

- Average Makespan: The average makespan over the horizon period is calculated as the average of the maximum completion times of surgeries for each day.

$$\sum_{d=1}^{D} \frac{Makespan_d}{D} \quad (12)$$

- Total Deviation: This measures the deviation from the initial schedule, calculated as the sum of absolute differences between the current and initial start times of surgeries.

$$Total\ Deviation = \sum_{i=1}^{N} \left(|Start\ Time_{i,current} - Start\ Time_{i,initial}|\right) \quad (13)$$

- Overlap Penalty: This penalty is added to the reward function to discourage scheduling conflicts, where multiple surgeries overlap in the same room or recovery bed.

$$Overlap\ Penalty = \sum_{r=1}^{R} \sum_{t=1}^{T} \max\ (0, Occupancy_r\ (t)\text{-}1) + \sum_{b=1}^{B} \sum_{t=1}^{T} \max\ (0, Occupancy_b\ (t)\ \text{-}1) \quad (14)$$

- The agent then updates the Q-values using the Bellman equation, where the Q-value for a given state-action pair is adjusted based on the immediate reward and the discounted future rewards. The Q-value update rule is:

$$Q\ (s, a) = Q\ (s, a) + \alpha[r + \gamma min a\ Q\ (s, a\ ) - Q\ (s\ , a\ )] \quad (15)$$

where α is the learning rate, γ is the discount factor, r is the immediate reward, and s' is the next state. This update mechanism allows the agent to iteratively improve its policy by considering both the immediate rewards and the potential future rewards, ensuring that the learned policy converges to an optimal solution over time.

4.5 Experience Replay and Target Networks

Experience replay is utilized to stabilize training by breaking the correlation between consecutive updates. The agent stores experiences in a replay buffer and samples mini-batches to update the Q-network. Additionally, target networks are used to provide stable target values for Q-value updates, further enhancing the training stability. The training process involves running multiple episodes, where the agent iteratively interacts with the environment, chooses actions, observes rewards, and updates the Q-table. The goal is to minimize the makespan and avoid overlaps in room and bed assignments while adapting to dynamic changes in the surgery schedule.

5 Numerical Results

In this section, we present the preliminary results of our proposed methodologies for both offline and online surgery planning and scheduling. The results are visualized to provide a clear understanding of the models' performance and their effectiveness in optimizing surgical schedules. Our experiments were conducted on a system with an Intel(R) Core (TM) i5-8265U CPU at 1.60GHz (max turbo 1.80 GHz) and 8GB RAM, running Windows 11 Home. We used Python (version 3.10) with the PuLP package for optimization and NumPy for data manipulation. Reinforcement learning was implemented using Python with custom RL libraries. For collaboration and code development, we utilized Google Colab and Anaconda Jupyter Notebook.

5.1 Dataset Description

The dataset was designed to mimic real-world healthcare scheduling scenarios. Surgical durations were modeled based on Landa [20], using real-world data and random selection.

Table.1. Instances Characterization

Instances	No. of patients	No. of ORs	No. of RBs	Planning Horizon	Start Time	Closing Time
P1-10-2-2-5	10	2	2	5	0	8
P2-10-3-2-3	10	3	2	3	0	8
P3-15-2-2-3	15	2	2	3	0	8
P4-15-3-3-5	15	3	3	5	0	10
P5-20-4-4-5	20	4	4	5	0	8

5.2 Parameter Settings

Our reinforcement learning (RL) model was designed to dynamically optimize surgical scheduling. Key parameters were:

- Learning Rate (0.9): Enables rapid adaptation to new information [21]
- Discount Factor (0.9): Balances immediate and long-term rewards [21]
- Initial Exploration Rate (0.5): Encourages diverse learning experiences.
- Exploration Decay Rate (0.05): Gradually shifts from exploration to exploitation.
- Minimum Exploration Rate (0.01): Ensures ongoing exploration.

These parameters were fine-tuned through iterative experimentation, allowing the RL model to effectively manage surgical scheduling complexities and improve efficiency in a dynamic healthcare environment.

5.3 Offline Results

We obtained an optimal offline solution for surgical scheduling using five different instances to establish a baseline for evaluating our RL model. Three instances, P1-10-2-2-5, P2-10-3-2-3, and P4-15-3-3-5 were detailed due to their variability. After running the MILP, we obtained the following schedules. The best way to present these results is through Gantt charts, which effectively illustrate the optimal allocation of surgeries to ORs and RBs, demonstrating the efficiency and effectiveness of our offline scheduling.

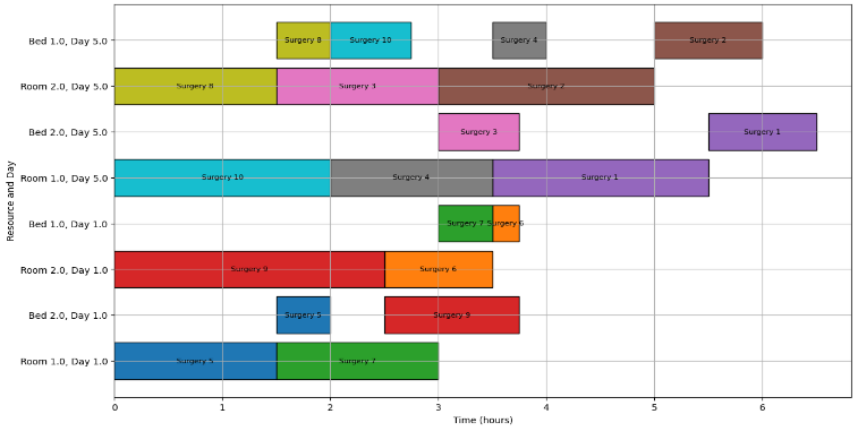

Fig. 4. Gantt Chart for Operating rooms and Recovery beds in Instance 1

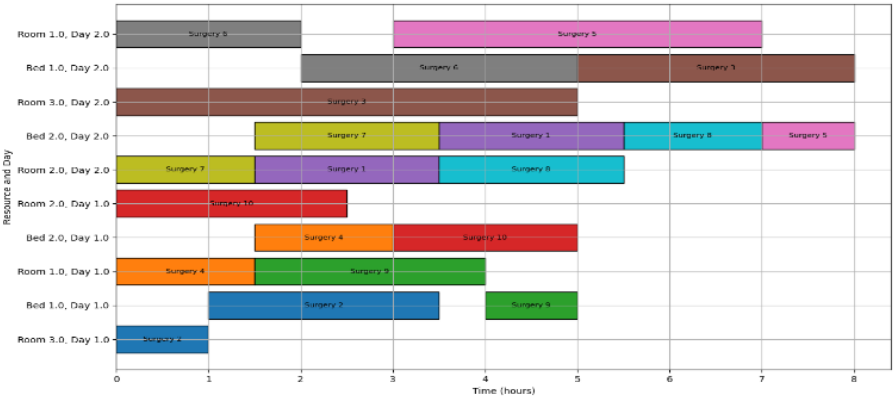

Fig. 5. Gantt Chart for Operating rooms and Recovery beds in Instance 2

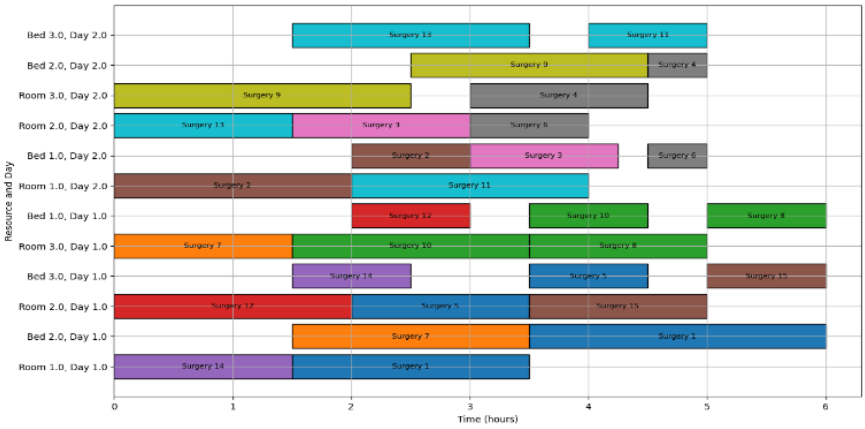

Fig. 6. Gantt Chart for Operating rooms and Recovery beds in Instance 3

Using a Mixed-Integer Linear Program (MILP) implemented through Python's PuLP package, the offline phase generates an initial static schedule that adheres to key constraints and objectives, ensuring feasibility and efficiency. This step, however, is computationally intensive and time-consuming, making it unsuitable for real-time decision-making. For this reason, offline scheduling is executed in the backend, where sufficient computational resources and time can be allocated. The resulting static schedule serves as an essential input for the subsequent real-time phase, providing a robust foundation that reinforcement learning (RL) leverages to enable quick reactions and dynamic adjustments.

5.4 Online Graphic Results

Following the offline phase, we simulated numerous disruptions such as surgery delays. The RL agent adjusted schedules in real-time, optimizing responses based on immediate feedback. The subsequent visualization illustrates the immediate schedule, highlighting disruption locations with arrows. The following Gantt chart demonstrates how the disruptions were managed. Additionally, a reward coverage line is included to illustrate how the RL model converges on the best actions that minimize both makespan and deviations.

Instance 1 Disruption: Surgery ID 5 experiences a delay of 1.5 hours

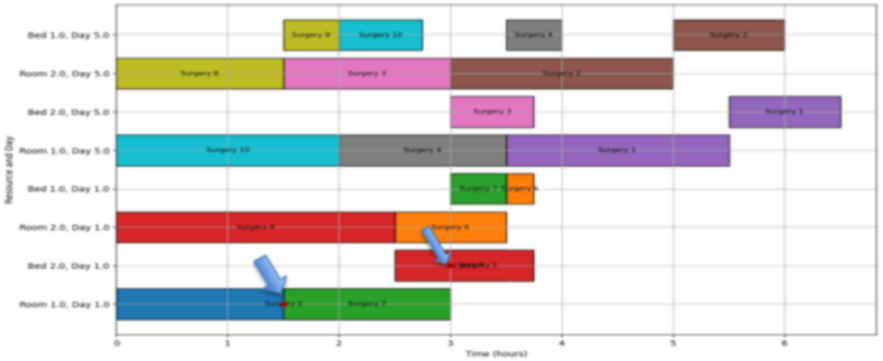

Fig. 7. Immediate Schedule after Disruption

=> RL Action: The RL model's action was to reschedule the surgery (Action = 1)

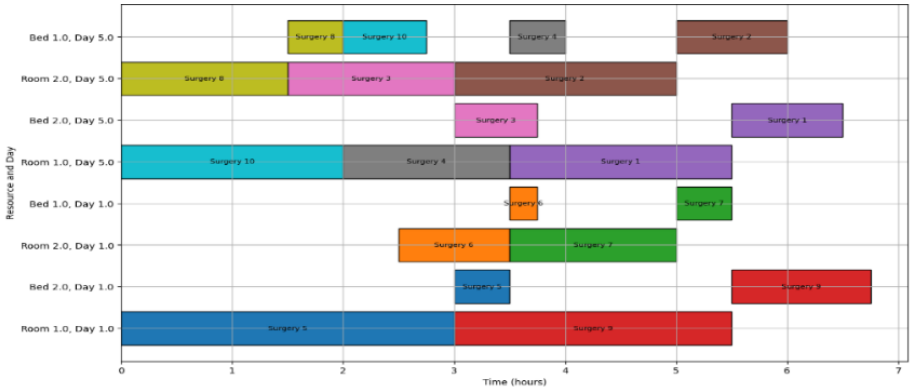

Fig. 8. Optimal State after RL Action on Disruption

Instance 1 Disruption: Reward Convergence

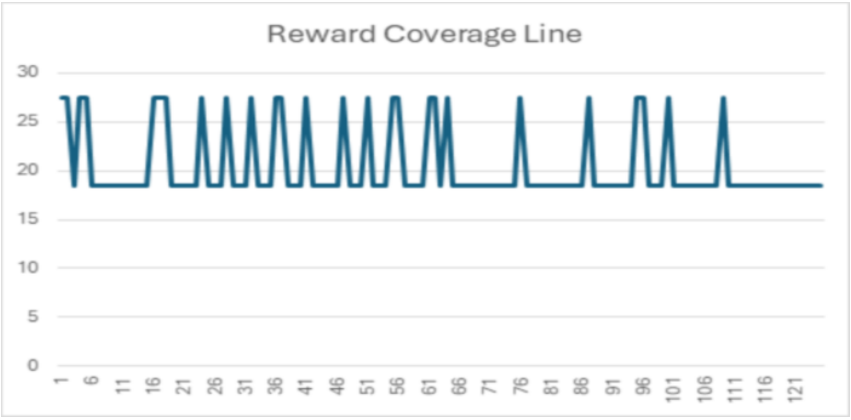

Fig. 9. RL Reward Coverage after Disruption

5.5 Online Numerical Results

In this section, we present the numerical results obtained from our reinforcement learning (RL) model applied to surgical scheduling. The table below summarizes the disruptions encountered, including the surgery ID affected, the delay duration, the RL model's execution time, and the actions taken by the RL model to address these disruptions. This table provides a clear overview of the model's performance and its responsiveness to various disruptions.

Table. 2. RL Implementation for Different Instances and Disruptions in Surgical Planning Scheduling

Instance	Disrupted Surgery ID	Delay (hours)	RL Agent Action	RL Execution Time (seconds)
Instance 1	ID 5	1.5	Action 1	0.20
	ID 6	7	Action 2	0.14
	ID 4	2.5	Action 1	0.16
Instance 2	ID 7	3.5	Action 1	0.17
	ID 4	2.5	Action 0	0.13
Instance 3	ID 7	1.5	Action 1	0.37
	ID 2	3	Action 1	0.25
Instance 4	ID 7	2.25	Action 1	0.14
Instance 5	ID 5	3	Action 0	0.27

Table 2. illustrates the RL model's efficiency in handling different surgical scheduling disruptions. Each instance represents a unique scenario with varying delays, highlighting the model's adaptability and responsiveness.

- Instance 1: The RL model effectively rescheduled surgeries for minor delays (e.g., 1.5 hours for ID 5) with minimal execution time (0.20 seconds). For a

significant delay (7 hours for ID 6), the model chose to do nothing, reflecting its ability to identify scenarios where rescheduling is unnecessary.

- Instance 2: The model consistently chose to reschedule surgeries for delays, demonstrating a quick response time, such as 0.17 seconds for a 3.5-hour delay in ID 7. This indicates the model's efficiency in minimizing disruptions by timely adjustments
- Instance 3: For more complex scenarios with multiple surgeries and significant delays, the RL model took slightly longer to execute (e.g., 0.37 seconds for a 1.5-hour delay in ID 7). This showcases the model's capability to handle increased complexity while still maintaining effective scheduling adjustments.
- Instance 4 and 5: The model's quick execution times (e.g., 0.14 seconds for a 2.25-hour delay in ID 7) and strategic actions like reassignment in Instance 5 demonstrate its robust performance across varied disruption scenarios.

Overall, the RL model exhibits a high level of efficiency and effectiveness in dynamically managing surgical schedules, ensuring optimal utilization of resources and minimizing the impact of delays. These results underscore the model's potential to improve operational efficiency in real-world healthcare settings

6 Conclusions and Future Works

This paper primarily focuses on tailoring reinforcement learning (RL) to address the integrated challenges of surgery planning and scheduling in operating rooms, a problem often studied in isolation within the literature. By combining a static schedule, developed using a Mixed-Integer Linear Program (MILP) via Python's PuLP package, with RL-driven dynamic adjustments, the proposed approach effectively accommodates real-time disruptions while maintaining adherence to the constraints and objectives of surgery scheduling. The static schedule serves as a foundation for RL to make adaptive adjustments, demonstrating the applicability of RL in healthcare settings where real-time responsiveness is critical.

Although time constraints and a lack of comparable frameworks in the literature limited direct benchmarking, future work should focus on incorporating comparisons with existing methods to further highlight the practical value of the proposed approach. Additionally, expanding the scope of this study to include variations in operating room and recovery bed durations, as well as additional resources such as surgical teams, will allow for more comprehensive modeling and testing of robustness.

References

1. Şeyda Gür, Tamer Eren, "Application of Operational Research Techniques in Operating Room Scheduling Problems: Literature Overview", Journal of Healthcare Engineering, vol. 2018, Article ID 5341394, 15 pages,2018
2. Xu, H., Fang, Y., Chou, C.A., Fard, N., & Luo, L. (2023). A reinforcement learning based optimal control approach for managing an elective surgery backlog after pandemic disruption. Health Care Management Science, 26, 430446.

3. Tsai, S. C., Yeh, Y., & Kuo, C. Y. (2021). Efficient optimization algorithms for surgical scheduling under uncertainty. European Journal of Operational Research, 293(2), 579593.
4. Miao, H., & Wang, J.J. (2021). Scheduling elective and emergency surgeries at shared operating rooms with emergency uncertainty and waiting time limit. Computers & Industrial Engineering, 160, 107551.
5. Breuer, D. J., Lahrichi, N., Clark, D. E., & Benneyan, J. C. (2020). Robust combined operating room planning and personnel scheduling under uncertainty. Volume, 27.
6. Allen, R. W., Taaffe, K., & Ritchie, G. (2015). Surgery rescheduling using discrete event simulation: A case study. *Pages*, 13651376.
7. Eshghali, M., Kannan, D., SalmanzadehMeydani, N., & Esmaieeli Sikaroudi, A. M. (2023). Machine learningbased integrated scheduling and rescheduling for elective and emergency patients in the operating theatre. Annals of Operations Research, 332, 9891012.
8. Varmazyar, M., AkhavanTabatabaei, R., Salmasi, N., & Modarres, M. (2019). Operating room scheduling problem under uncertainty: Application of continuous phasetype distributions. Scheduling & Logistics, 216235.
9. Saadouli, H., Jerbi, B., Dammak, A., Masmoudi, L., & Bouaziz, A. (2015). A stochastic optimization and simulation approach for scheduling operating rooms and recovery beds in an orthopedic surgery department. Computers & Industrial Engineering, 80, 72–79.
10. Doulabi, H. D., & Khalilpourazari, S. (2023). Stochastic weekly operating room planning with an exponential number of scenarios. Annals of Operations Research, 328, 643–664.
11. Kamran, M. A., Karimi, B., & Dellaert, N. (2018). Uncertainty in advance scheduling problem in operating room planning. Computers & Industrial Engineering, 126, 252268.
12. Lamiri, M., Xie, X., Dolgui, A., & Grimaud, F. (2008). A stochastic model for operating room planning with elective and emergency demand for surgery. Volume, 185(5), 10261037.
13. Yu, H., Gao, K., Wu, N., Zhou, M., Suganthan, P. N., & Wang, S. (2024). Scheduling Mult objective dynamic surgery problems via Q-learning based metaheuristics. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 54(6).
14. Silva, A. B., Gonçalves, J. F., & Relvas, S. (2019). Surgical scheduling under uncertainty using approximate dynamic programming. Computers & Operations Research, 98, 114.
15. Spangenberg, N., Wilke, M., Augenstein, C., & Franczyk, B. (2018). Online surgery rescheduling A datadriven approach for realtime decision support. Pages, 336343.
16. Kamran, M. A., Karimi, B., & Dellaert, N. (2018). Uncertainty in advance scheduling problem in operating room planning. Computers & Industrial Engineering, 126, 252268.
17. Beaulieu, M., Gendreau, M., & Soriano, P. (2012). Operating rooms scheduling under uncertainty. Pages, 1332.
18. Xu, H., Fang, Y., Chou, C.A., Fard, N., & Luo, L. (2023). A reinforcement learningbased optimal control approach for managing an elective surgery backlog after pandemic disruption. Health Care Management Science, 26, 430446
19. Eshghali, M., Kannan, D., SalmanzadehMeydani, N., & Esmaieeli Sikaroudi, A. M. (2023). Machine learningbased integrated scheduling and rescheduling for elective and emergency patients in the operating theatre. Annals of Operations Research, 332, 9891012.
20. Landa, P., Aringhieri, R., Soriano, P., Tànfani, E., & Testi, A. (2016). A hybrid optimization algorithm for surgeries scheduling. Operations Research for Health Care, 8, 103114.
21. Zhu, S., Fan, W., Yang, S., Pei, J., & Pardalos, P. M. (2019). Operating room planning and surgical case scheduling: a review of literature. Journal of Combinatorial Optimization, 37, 757805