Optimizing Hospital Waiting Lists using Reinforcement Learning

Daniele Cecca
Matr. 914358
MSc Artificial Intelligence for Science and Technology
Email: d.cecca@campus.unimib.it

***Abstract*—Efficient management of hospital waiting lists is a critical operational and clinical challenge. Limited resources, stochastic patient arrivals, and progressive clinical deterioration make scheduling decisions highly complex. In this work, we formulate waiting list management as a Markov Decision Process and propose a reinforcement learning approach to learn optimal scheduling policies. We compare Maskable Proximal Policy Optimization and Deep Q-Learning against a FIFO and TOP-K baseline**

1. Introduction

Hospital waiting list management plays a crucial role in modern healthcare systems. Suboptimal scheduling decisions may lead to excessive waiting times, increased clinical risk, and inefficient use of hospital capacity. Traditional scheduling strategies, such as First-In-First-Out (FIFO) or fixed priority rules, are typically static and fail to capture the dynamic nature of patient deterioration and arrivals.

Reinforcement Learning (RL) offers a principled framework for sequential decision-making under uncertainty. By modeling the problem as a Markov Decision Process (MDP), an agent can learn scheduling policies that explicitly account for future consequences of present decisions.

In this work, we investigate the application of RL to hospital waiting list optimization and compare two deep RL algorithms: Proximal Policy Optimization (PPO) and Deep Q-Learning (DQN)

2. Environment Definition

2.1. Markov Decision Processes

A Markov Decision Process is defined by the tuple

$$(S, A, P, R, \gamma),$$

where S denotes the state space, A the action space, P the transition probability function, R the reward function, and $\gamma \in [0, 1]$ the discount factor.

The Markov property assumes that the future evolution of the system depends only on the current state and action.

2.1.1. State Space. The environment state provides a structured representation of both the hospital waiting list and the current operational conditions. At each decision step, only the top-K patients in the queue are observed to ensure scalability.

Each patient $i \in \{1, \dots, K\}$ is described by the feature vector

$$p_i = (r_i,\ t_i,\ g_i,\ u_i,\ d_i)\,,$$

where:

- $r_i \in [0, 1]$ denotes the current clinical risk of patient i,
- $t_i \in \mathbb{N}_0$ is the waiting time (in days),
- $g_i \in [0, g_{\max}]$ represents the daily deterioration rate,
- $u_i \in \{0, 1, 2, 3\}$ is the urgency class,
- $d_i \in \mathbb{N}$ is the intervention duration (in this work fixed to $d_i = 1$).

In addition to patient-level information, the state includes a set of global hospital-level variables

$$h = (c_t,\ q_t,\ \tau_t)\,,$$

where:

- $c_t \in \{0, \dots, C\}$ is the remaining daily capacity,
- $q_t \in \mathbb{N}_0$ denotes the total queue length,
- $\tau_t \in \{0, \dots, T_{\max}\}$ is the current day within the planning horizon.

The full state is therefore given by

$$s_t = (p_1, \dots, p_K,\ h)\,.$$

The state is represented as a dictionary observation, separating patient-level and global features. This structured representation enables the use of multi-input neural policies, where each component is encoded by a dedicated neural network module.

2.1.2. Action Space. At each timestep t, the agent selects one patient to schedule among the observed patients. The action space is defined as

$$A = \{0, 1, \dots, K - 1\},$$

where action $a_t = i$ corresponds to scheduling patient i.

If fewer than K patients are present in the queue, actions referring to nonexistent patients are considered invalid. Such actions are excluded through an action masking mechanism, ensuring that the policy assigns zero probability to invalid choices.

2.1.3. Transition Dynamics. The transition function models the daily operational dynamics of the hospital. Each selected action consumes one unit of daily capacity. When the remaining capacity reaches zero or the queue becomes empty, the current day ends.

For each patient i remaining in the queue at the end of day t, the state evolves; When patient i remains in the queue for Δt days without intervention, risk updates as

$$r_i(t + \Delta t) = \Pi_{[0,1]}\big(r_i(t) + g_i\,\Delta t\big),$$

New patient arrivals occur at the end of each day according to a Poisson process:

$$N_t \sim \text{Poisson}(\lambda),$$

where λ denotes the average daily arrival rate. At the beginning of the next day, the daily capacity is reset to its maximum value C.

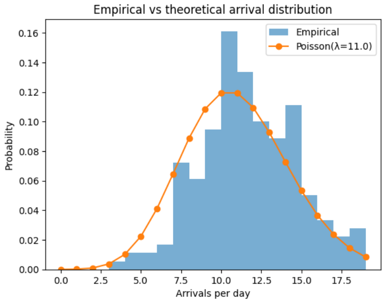


Figure 1: Poisson distribution

In the proposed environment, a single timestep corresponds to the scheduling of one patient, rather than to the progression of a full day. Multiple timesteps are therefore executed within the same simulated day until the available daily capacity is exhausted or the waiting list becomes empty. An episode consists of a fixed number of simulated days, after which the environment is truncated. As a result, each episode spans several thousand decision steps, yielding a long-horizon reinforcement learning problem with sparse, delayed rewards.

2.1.4. Reward Function. The reward is computed at the end of each day and is defined as the negative of a cost function that captures both clinical and operational objectives.

Let Δr_t denote the cumulative increase in clinical risk across all waiting patients, L_t the number of patients whose waiting time exceeds a predefined threshold τ_{max}, and U_t the number of unused capacity slots. The daily reward is defined as:

$$R_t = -\left(\Delta r_t + w_{\text{late}} \cdot L_t + w_{\text{unused}} \cdot U_t\right),$$

where w_{late} and w_{unused} are weighting coefficients.

This reward formulation encourages policies that minimize long-term clinical deterioration, avoid excessive waiting times, and efficiently utilize hospital resources.

2.2. Patient Parameter Generation

2.2.1. Urgency class. Upon arrival, each patient i is assigned an urgency class

$$u_i \sim \text{Unif}\{0, 1, 2, 3\},$$

where larger values represent more urgent clinical conditions. This choice induces a balanced mix of severities in the synthetic population.

2.2.2. Baseline risk at arrival. Given u_i, we sample a baseline arrival risk using a class-dependent mean plus Gaussian noise:

$$\tilde{r}_i = \mu_r(u_i) + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0, \sigma_r^2),$$

with

$$\mu_r(u) = [0.05,\ 0.10,\ 0.20,\ 0.35]_u, \qquad \sigma_r = 0.02.$$

Since risk is modeled as a bounded quantity, we enforce

$$r_i(0) = \Pi_{[0,1]}(\tilde{r}_i),$$

where $\Pi_{[0,1]}(\cdot)$ denotes projection onto $[0, 1]$ (implemented via clipping). This construction produces partially overlapping risk distributions across urgency classes, avoiding deterministic separation and reflecting realistic within-class heterogeneity.

2.2.3. Daily deterioration rate. Each patient is also assigned an individual daily deterioration rate g_i that controls how quickly risk increases while waiting. We draw

$$\tilde{g}_i = \mu_g(u_i) + \eta_i, \qquad \eta_i \sim \mathcal{N}(0, \sigma_g^2),$$

with

$$\mu_g(u) = 0.01 + 0.01\,u, \qquad \sigma_g = 0.005,$$

and enforce boundedness through

$$g_i = \Pi_{[0,g_{\text{max}}]}(\tilde{g}_i), \qquad g_{\text{max}} = 0.10.$$

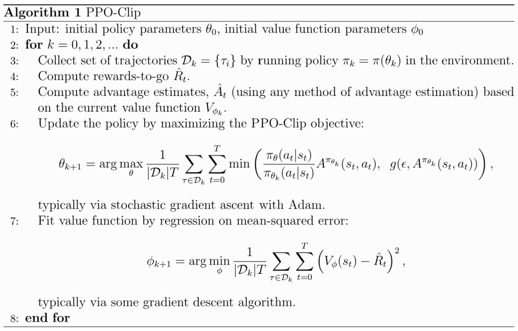
Hence, higher urgency classes are expected to deteriorate faster on average (larger μ_g), while still allowing for variability and occasional overlap among classes. The cap g_{max} prevents unrealistically fast deterioration trajectories that would dominate the dynamics.

3. Maskable Proximal Policy Optimization

3.1. Theoretical Background

Proximal Policy Optimization is an on-policy actor-critic algorithm designed to ensure stable policy updates. PPO optimizes a clipped surrogate objective that limits the change in policy probabilities between consecutive updates. PPO learns a stochastic policy while simultaneously training a value function used to estimate advantages and stabilize policy updates.

To handle invalid actions, a maskable variant of PPO is employed, which enforces zero probability for invalid actions during both training and evaluation.



```
Algorithm 1 PPO-Clip
Input: initial policy parameters $\theta_0$, initial value function parameters $\phi_0$
for $k = 0, 1, 2, ...$ do
    Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
    Compute rewards-to-go $\hat{R}_t$.
    Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function $V_{\phi_k}$.
    Update the policy by maximizing the PPO-Clip objective:
        $\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t),\ \ g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$
    typically via stochastic gradient ascent with Adam.
    Fit value function by regression on mean-squared error:
        $\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left( V_\phi(s_t) - \hat{R}_t \right)^2,$
    typically via some gradient descent algorithm.
end for
```

Figure 2: Pseudo-code PPO

3.2. Training

The policy network uses a multi-input architecture based on multilayer perceptrons, processing patient-level and global features separately before merging them. Key training parameters include a high discount factor ($\gamma = 0.99$), long rollout horizons, entropy regularization for exploration, and clipping of policy updates for stability.

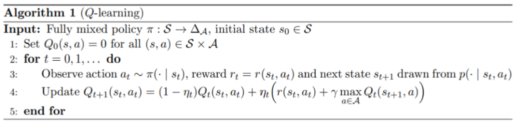
TABLE 1: Training hyperparameters for Maskable Proximal Policy Optimization (PPO).

Parameter	Value
Policy architecture	MultiInputPolicy (MLP-based)
Algorithm	Maskable PPO
Discount factor γ	0.99
Learning rate	3×10^{-4}
Rollout length (n_{steps})	2048
Batch size	256
Number of epochs per update	10
GAE parameter λ	0.95
Clipping range ϵ	0.2
Entropy coefficient	0.01
Value function coefficient	0.5
Max gradient norm	0.5
Action masking	Enabled
Total training timesteps	200,000

4. Deep Q-Learning

4.1. Theoretical Background

Deep Q-Learning is an off-policy value-based algorithm that approximates the action-value function $Q(s, a)$ using a deep neural network. The network is trained by minimizing the Bellman error using samples drawn from a replay buffer. Stability is improved through the use of a target network and an ϵ-greedy exploration strategy.



```
Algorithm 1 (Q-learning)
Input: Fully mixed policy $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$, initial state $s_0 \in \mathcal{S}$
Set $Q_0(s, a) = 0$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$
for $t = 0, 1, \ldots$ do
    Observe action $a_t \sim \pi(\cdot \mid s_t)$, reward $r_t = r(s_t, a_t)$ and next state $s_{t+1}$ drawn from $p(\cdot \mid s_t, a_t)$
    Update $Q_{t+1}(s_t, a_t) = (1 - \eta_t)Q_t(s_t, a_t) + \eta_t\Big(r(s_t, a_t) + \gamma \max_{a \in \mathcal{A}} Q_t(s_{t+1}, a)\Big)$
end for
```

Figure 3: psudo-code q-learning

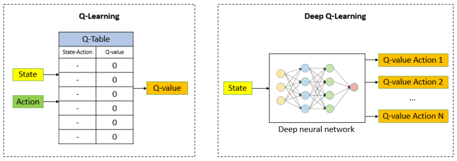


Figure 4: DQN VS QL

4.2. Training

TABLE 2: Training hyperparameters for Deep Q-Learning (DQN).

Parameter	Value
Policy architecture	MultiInputPolicy (MLP-based)
Algorithm	DQN
Discount factor γ	0.99
Learning rate	1×10^{-4}
Replay buffer size	100,000
Batch size	256
Training frequency	Every 4 environment steps
Target network update interval	10,000 steps
Exploration strategy	ε-greedy
Initial exploration $\varepsilon_{\text{start}}$	1.0
Final exploration $\varepsilon_{\text{final}}$	0.05
Exploration fraction	0.2
Learning starts	10,000 steps
Action masking	Not used (penalty-based)
Total training timesteps	300,000

4.3. Multi-Input Policy Architecture.

To handle the structured state representation, we employ a multi-input policy architecture. Patient-level features and global hospital-level variables are processed by separate neural network encoders, each implemented as a multilayer perceptron. The resulting latent representations are concatenated and used to estimate both the policy and the value function. This design preserves the semantic structure of the state and allows the agent to jointly reason over individual patient characteristics and global operational conditions.

5. Evaluation

The evaluation is carried out at two different temporal scales in order to assess both the global and the fine-grained behavior of the proposed scheduling policy.

Episode-level (macroscopic) evaluation

At a macroscopic level, we analyze the system performance over complete episodes. In this setting, each episode spans the full simulation horizon, and the results are averaged over 30 independent episodes in order to reduce stochastic variability.

The following metrics are considered:

- **Return.** The cumulative reward obtained over an episode. This metric summarizes the overall performance of the policy by combining clinical deterioration, service-level violations, and resource utilization into a single scalar value. Higher values indicate better overall performance.
- **ΔRisk sum.** The total increase in patient risk accumulated during the episode. This metric captures the global clinical deterioration of patients while waiting in the queue. Lower values indicate better clinical outcomes.
- **Late count.** The total number of patients whose waiting time exceeds the predefined threshold τ_{wait}. This metric reflects violations of service-level agreements (SLAs) or clinical timeliness constraints. Lower values are preferable.
- **Invalid actions.** The number of invalid scheduling actions selected by the agent. An invalid action corresponds to choosing a non-existent or masked patient. This metric measures the agent's ability to respect feasibility constraints. Lower values indicate better policy stability.
- **Mean queue length.** The average number of patients in the waiting list throughout the episode. This metric provides an indication of system congestion and backlog management.

For each metric, we report the mean value and the standard deviation across episodes.

Day-level (microscopic) evaluation

At a microscopic level, we analyze the day-by-day dynamics of the system over a time window of 60 days. This analysis provides insight into transient behaviors, stability, and trade-offs that are not visible from episode-level aggregates.

The following daily metrics are monitored and visualized:

- **Queue length.** The number of patients waiting in the queue at each day. This metric reflects system congestion and backlog evolution over time.
- **Used capacity.** The number of scheduling slots effectively used during the day. It measures how efficiently the available daily capacity is exploited.
- **Unused slots.** The number of available slots left unused at the end of the day. This metric captures inefficiencies in resource utilization.
- **Daily ΔRisk.** The total increase in patient risk accumulated during a single day. It provides a fine-grained view of clinical deterioration dynamics.
- **Late patients (per day).** The number of patients exceeding the waiting time threshold on a given day. This metric highlights daily SLA or timeliness violations.
- **Mean and maximum risk in queue.** The average and maximum patient risk observed in the queue at the end of each day. These metrics characterize both typical and worst-case clinical severity levels.
- **Mean waiting time.** The average waiting time (in days) of patients remaining in the queue at the end of the day. This metric quantifies access delays experienced by patients.
- **Urgency mix in queue.** The distribution of patients across urgency classes (urg $= 0, \ldots, 3$) at the end of each day. This analysis shows how the scheduling policy prioritizes different urgency levels over time.

Together, the macroscopic and microscopic evaluations provide a comprehensive assessment of the policy, capturing both long-term performance and short-term operational dynamics.

5.1. PPO evaluation

PPO attains an average return of -479.072 ± 149.152. The cumulative risk increase is 343.994 ± 96.396, which is the lowest among the two learned policies, indicating that PPO is particularly effective at limiting long-term clinical deterioration. However, PPO exhibits a substantially larger number of late patients, 2701.567 ± 1068.279, suggesting that this risk reduction is achieved at the expense of timeliness/SLA violations. No invalid actions are observed, confirming that feasibility is respected during evaluation.

Metric	**Value (mean ± std)**	**Notes**
Episodes	30	–
Return	-479.072 ± 149.152	–
ΔRisk sum	343.994 ± 96.396	Lower is better
Late count	2701.567 ± 1068.279	Lower is better
Invalid	0.000 ± 0.000	Lower is better
Queue mean	132.317 ± 31.013	–

TABLE 3: Evaluation results over 30 episodes.

The day-level trajectories suggest a policy that prioritizes risk mitigation: risk-related curves evolve in a comparatively controlled manner, without persistent saturation of maximum risk. At the same time, late-patient curves show earlier and/or more frequent SLA violations than DQN, consistent with the high episode-level late count. The queue length grows gradually over time, reflecting the structural load of the system rather than short-term instability. Overall, PPO appears to implement a more risk-driven selection strategy, which can leave some patients waiting beyond the threshold when capacity is tight, thereby increasing lateness while keeping global risk accumulation lower..

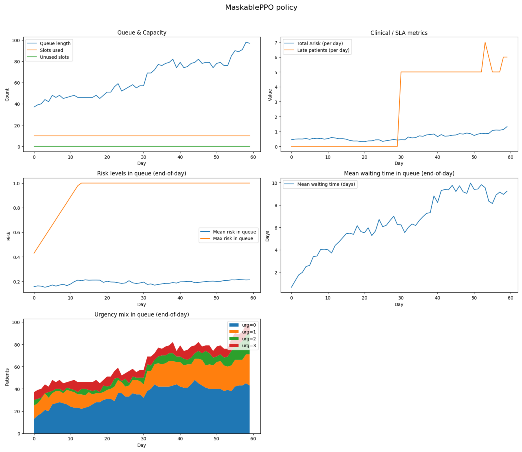


Figure 5: ppo stats

5.2. DQN evaluation

DQN achieves an average return of -454.629 ± 139.490, improving over PPO in terms of the aggregated reward. The cumulative risk increase is 398.355 ± 101.922, higher than PPO, indicating weaker performance in limiting clinical deterioration. In contrast, the late patient count is markedly lower, 1125.467 ± 798.763, showing that DQN is substantially better at preventing threshold violations. As for PPO, the invalid action rate is 0.000 ± 0.000.

Metric	**Value (mean ± std)**	**Notes**
Episodes	30	–
Return	-454.629 ± 139.490	–
ΔRisk sum	398.355 ± 101.922	Lower is better
Late count	1125.467 ± 798.763	Lower is better
Invalid	0.000 ± 0.000	Lower is better
Queue mean	132.317 ± 31.013	–

TABLE 4: Evaluation results over 30 episodes.

The daily plots are consistent with a policy that is more timeliness-oriented: late patients appear later and/or in smaller daily amounts compared to PPO, in line with the lower episode-level late count. However, risk-related curves display larger accumulation and more pronounced peaks, indicating that high-risk patients may remain in the queue longer, with risk being "paid" to keep waiting times below the SLA threshold. Queue length evolution remains increasing but does not show the extreme instability typical of FIFO. Overall, DQN appears to trade clinical risk for improved timeliness, yielding fewer late patients but higher cumulative deterioration.

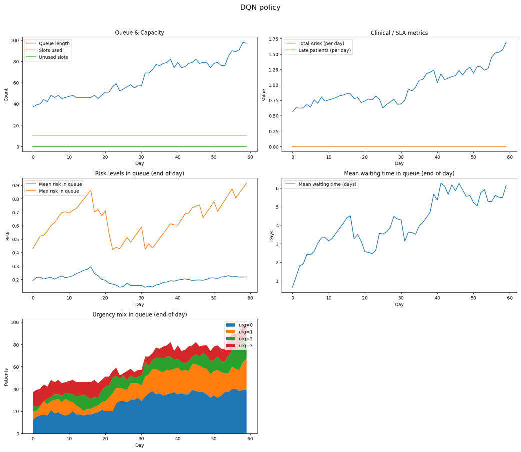


Figure 6: dqn stats

5.3. Comparison PPO, DQN, Top-K, and FIFO

Figures 7–8 and Tables 4–6 highlight a clear trade-off between clinical risk minimization and SLA compliance.

FIFO remains the weakest baseline, with rapid congestion growth and severe deterioration dynamics: day-level plots show saturation of maximum risk and a sharp escalation of late patients after a critical point, confirming that strict arrival-order scheduling is inadequate under heterogeneous deterioration.

The Top-K heuristic improves substantially over FIFO by explicitly prioritizing high-risk patients, yielding low cumulative risk and comparatively limited lateness. Nevertheless, it remains a static rule and cannot adapt to changes in queue composition and stochastic arrivals, leading to steady queue growth and increasing waiting times.

Comparing the two RL agents under the corrected results, PPO achieves the lowest cumulative risk increase but suffers from a much larger late count. DQN, instead, reduces SLA violations substantially (lower late count) but accumulates more clinical risk. Hence, PPO behaves as a risk-minimizing policy, whereas DQN behaves as a timeliness-preserving policy under the current reward weights.

Overall, neither learned policy dominates across all objectives: PPO provides better clinical-risk control, while DQN provides better SLA compliance. The preferred method therefore depends on the relative importance assigned to deterioration versus lateness (i.e., on the reward weights w_{late} and the implicit cost of risk accumulation).

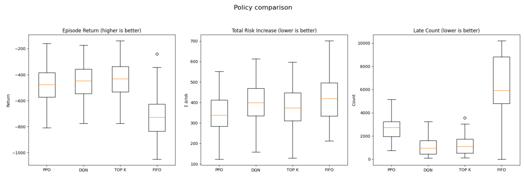


Figure 8: comparisons episodes stats

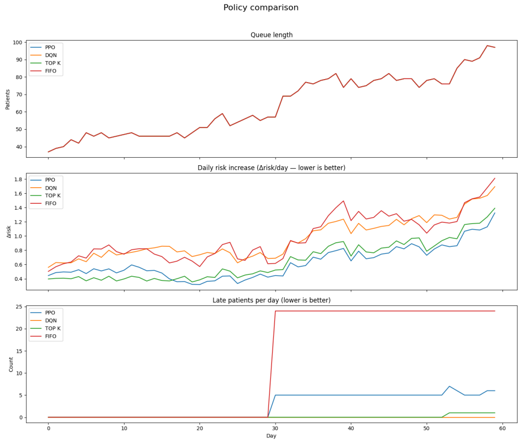


Figure 7: comparisons daily stats

6. Conclusion

In this work, hospital waiting list management was formulated as a long-horizon sequential decision-making problem under uncertainty. A stochastic simulation environment was introduced to capture key operational and clinical aspects of real healthcare systems, including limited daily capacity, random patient arrivals, heterogeneous deterioration rates, and service-level constraints on waiting times. The problem was modeled as a Markov Decision Process with structured observations and delayed, clinically meaningful rewards.

Two deep reinforcement learning approaches were evaluated: Maskable Proximal Policy Optimization (PPO) and Deep Q-Learning (DQN), and compared against FIFO and Top-K heuristic baselines. The results clearly demonstrate the importance of risk-aware scheduling. FIFO consistently performed worst, leading to rapid queue congestion, severe clinical deterioration, and a large accumulation of late patients, confirming that static arrival-order policies are inadequate in settings with heterogeneous patient deterioration.

The Top-K heuristic substantially improved clinical outcomes by prioritizing high-risk patients, but lacked adaptivity to stochastic demand and evolving queue composition, resulting in steady queue growth and limited long-term robustness.

Among the learned policies, PPO and DQN exhibited complementary behaviors. PPO achieved the lowest cumulative risk increase, indicating effective control of clinical deterioration, but incurred a large number of late patients, suggesting a strong emphasis on risk minimization over timeliness. Conversely, DQN significantly reduced SLA violations, yielding a much lower late count, but at the cost of higher cumulative risk. These results highlight an intrinsic trade-off between clinical risk control and waiting-time compliance, which is strongly influenced by the reward formulation.

Overall, the findings suggest that reinforcement learning can learn meaningful and non-trivial scheduling strategies that outperform static heuristics, but that no single policy is universally optimal across all objectives. Future work should focus on multi-objective or constrained reinforcement learning formulations to explicitly balance clinical risk and timeliness, as well as on extending the environment toward higher fidelity by incorporating variable intervention durations, multiple resources, and data-driven risk estimation models. Despite its abstractions, the proposed framework provides a reproducible testbed and a clear empirical demonstration of the potential of reinforcement learning for adaptive, clinically motivated hospital scheduling.

7. More analysis

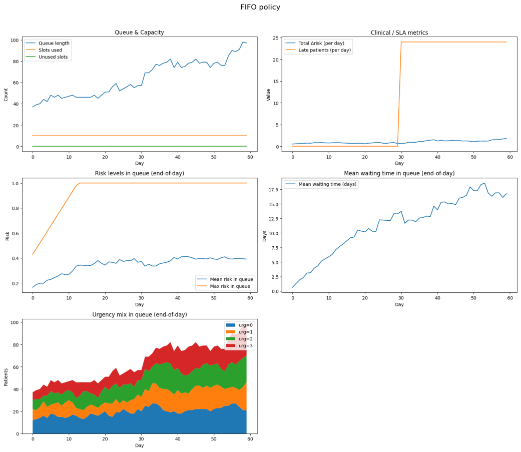


Figure 9: Fifof episodes

Metric	Value (mean ± std)	Notes
Episodes	30	–
Return	-720.504 ± 180.078	–
ΔRisk sum	415.052 ± 114.096	Lower is better
Late count	6109.033 ± 2755.161	Lower is better
Invalid	0.000 ± 0.000	Lower is better
Queue mean	132.317 ± 31.013	–

TABLE 5: Evaluation results over 30 episodes.

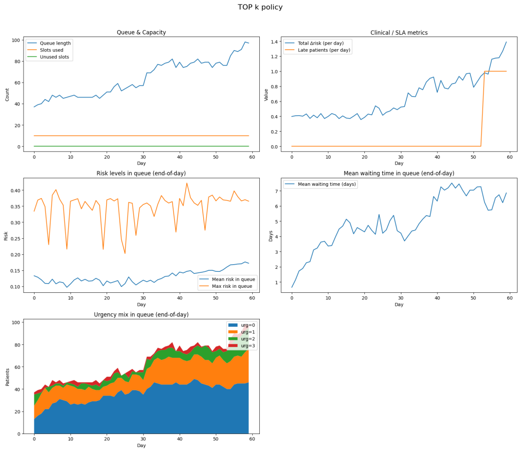


Figure 10: Top K episodes

Metric	Value (mean ± std)	Notes
Episodes	30	–
Return	-437.329 ± 147.773	–
ΔRisk sum	373.795 ± 106.089	Lower is better
Late count	1270.667 ± 874.122	Lower is better
Invalid	0.000 ± 0.000	Lower is better
Queue mean	132.317 ± 31.013	–

TABLE 6: Evaluation results over 30 episodes.

References

.