Single- and Multi-Agent Approaches to Equitable MELD-Driven Liver Allocation

CS238 Fall 2024: Decision Making Under Uncertainty

Elsa Bismuth, Justine Breuch, Eden Luvishis, Veronica Yakubovich

December 6, 2024

Abstract

This study addresses the challenge of liver transplant allocation, which requires balancing equity and patient survival under resource constraints. We first offer an approach that optimizes patient and graft lifespan, and then redefine the allocation process as a multi-agent stochastic game, where three patient risk groups—low, medium, and high—compete for organs using Nash Q-learning. Our contributions are the following: (1) a DQN-derived policy framework for maximizing life and transplant duration (2) a Nash Q-learning model, treating each risk group as an independent agent competing for resources, introducing strategic interactions; (3) we replace synthetic urgency distributions with real MELD score distributions, derived using advanced statistical modeling for clinical realism; (4) we achieve fairness under the Nash Q-learning regime via equilibrium, ensuring no group can unilaterally improve its allocation without adjusting to others; Results demonstrate similar lifespan performance to United Network of Organ Sharing (UNOS) through DQN, while the group-based framework effectively promotes equity across risk groups while maintaining high survival outcomes, highlighting its potential for developing equitable and effective organ allocation policies.

Introduction

Liver transplant allocation is a critical decision-making problem that requires balancing patient survival and equity under severe organ shortages. Current allocation policies, governed by the United Network of Organ Sharing (UNOS), prioritize candidates based on the Model for End-Stage Liver Disease (MELD) score, which estimates short-term mortality risk. We use Deep Q-Networks [12] to model an approach that optimizes patient and graft lifespan. Then, instead of prioritizing individual patients, we address fairness across risk groups using Nash-Q learning and model dynamic trade-offs between urgency and organ availability.

To address these limitations, we first consider allocation on a per-organ per-patient basis through a single-agent Deep Q-Network, without geography or equity constraints. This optimizes the patient and transplant (graft) lifespan while requiring blood-type matches. Then we use a data-driven approach to redefine the organ allocation problem as a multi-agent stochastic game. By leveraging Nash Q-learning, we model interactions among three competing risk groups—low, medium, and high—and aim to achieve equitable outcomes at equilibrium. Our method introduces real MELD score distributions, enhancing clinical realism and providing a stronger foundation for decision-making. The proposed framework offers a significant advancement over existing single-agent Markov Decision Process (MDP) approaches by decentralizing decisions and promoting fairness through competitive dynamics.

Related Work

Organ allocation for liver transplants is currently managed by the United Network of Organ Sharing (UNOS), which uses a computer-based system incorporating factors such as blood type, medical urgency, and proximity to donors [1]. For livers, medical urgency is quantified using the Model for End-Stage Liver Disease (MELD) score [2], a clinical metric based on serum bilirubin, INR, and creatinine levels, with higher scores representing higher medical urgency. Despite its effectiveness, the system faces challenges in addressing inequities across patient groups, as highlighted in prior studies [3].

Healthcare operations research has explored optimization approaches to improve organ allocation policies. Markov Decision Processes (MDPs) and Partially Observable MDPs (POMDPs) have proven effective for modeling sequential decision problems under uncertainty [4]. For example, MDPs have been used in kidney transplantation to balance organ availability and patient health dynamics [5], while POMDPs have addressed incomplete waiting list information, achieving transparency and fairness trade-offs [6]. In liver allocation, [7] applied Q-learning to optimize allocation decisions, balancing urgency and allocation. However, their reliance on synthetic urgency distributions and a single-agent MDP framework limited clinical realism and equity. To the best of our knowledge, there is not prior work using Deep Q-Networks for organ allocation.

Recent work has emphasized the importance of using real MELD score distributions to model patient urgency. For instance, [8] identified distinct statistical patterns in MELD scores across risk groups, showing

that Gamma, Beta, and Exponential distributions provide robust fits. Advanced statistical techniques, such as the Kolmogorov-Smirnov method, further enhance the accuracy of such models [9].

Multi-agent frameworks offer a promising alternative to single-agent MDPs. The coordinated MDP approach proposed by [10] demonstrated the benefits of decentralized decision-making for resource allocation, modeling agents that balance individual utilities and global outcomes. By integrating Nash Q-learning into the liver allocation process, we aim to extend these ideas to a healthcare context, enabling equitable competition among patient groups while leveraging real-world urgency data.

Dataset and Features

Dataset

We gained access to the STAR (Standard Transplant Analysis and Research) to inform our models and experiment setup. The dataset includes liver transplants (and outcomes), waiting lists, and donor data since 1987 [11]. The transplant data specifically proved most valuable as it provided statistical grounding for the parameters in the Multi-Agent Markov Decision Process (MMDP) model and provided training data for the Deep Q-Network (DQN). Each column in the datasets was encoded with specific values set by STAR, therefore, we first remapped the data to integer values and filtered for relevant data within each column. For the model training, we chose to limit our data to only deceased donors, adult recipients, and those who had either received a transplant or died waiting for a transplant. This results in 139,000 rows. The MELD, or health severity scores, produced empirical distributions for the MMDP Nash-Q model and informed the list of actions available to the DQN. The DQN also leveraged several patient features in modeling the states and actions.

Methodology

Multi-Agent Markov Decision Process (MMDP)

Building on [7], which modeled organ allocation as a Multi-Agent Markov Decision Process (MMDP) optimized via Q-learning, we extend the framework as follows:

1. **Nash Q-learning**: Rather than a single central decision-maker, we categorize patients in 3 risk groups: Low-Risk (LR), Medium-Risk (MR), High-Risk (HR). Each group is treated as an independent agent learning its own priority threshold. This decentralization introduces strategic interactions in auction-like form.
2. **Data-driven Urgencies**: We replace synthetic urgency distributions with empirical distributions derived from real MELD scores, enhancing the clinical realism of the model.
3. **Fairness via Equilibrium**: By seeking a Nash equilibrium, we achieve an allocation where no single group can unilaterally improve its share, promoting more equitable outcomes.

This section details the environment formulation, urgency modeling, patient cohorts, learning framework, and implementation for both the single and multi-agent models.

Patient Cohort and Risk Grouping

We consider a cohort derived from a large-scale registry of liver transplant candidates, totaling over 113,000 patients. After filtering and ensuring the availability of initial MELD lab scores, we classify patients into three risk groups based on their MELD score distribution:

- Low-Risk Group (LR): MELD ≤ 9, totaling 15,467 patients.
- Medium-Risk Group (MR): $9 <$ MELD < 40, totaling 91,771 patients.
- High-Risk Group (HR): MELD ≥ 40, totaling 6,367 patients.

This grouping captures clinically relevant stratifications of patient severity and urgency. Our analysis leverages these groupings to assign organs and measure equity across risk profiles.

State Space

We consider a discrete-time process over a 30-day horizon with an initial organ supply $O_0 = 50$. At each time step t, the state is:

$$s_t = \big(t,\, O_t,\, (m_{LR}, m_{MR}, m_{HR})\big),$$

where O_t is the remaining number of organs, and m_g is the number of organs allocated to group $g \in$ {LR,MR,HR} at the previous step. This representation captures temporal progression, resource availability, and recent allocation patterns as established by [7].

Action Space

At each time step, an action corresponds to selecting a priority threshold vector:

$$a_t = (a_{LR}, a_{MR}, a_{HR}), \quad a_g \in \{0, \ldots, 10\}.$$

These thresholds determine which patients within each group are deemed sufficiently urgent to receive organs if available.

Transition Dynamics

We sample a small number of patients from each group per day (e.g., LR:1, MR:2, HR:5). Each patient's urgency u is drawn from fitted distributions (see below). If $u > a_g$ and $O_t > 0$, an organ is allocated. The next state updates:

$$O_{t+1} = O_t - \sum_g m_g,$$

and (m_{LR}, m_{MR}, m_{HR}) is set to reflect the newly allocated organs. This captures the resource consumption and evolving patient arrival dynamics.

Reward Function

In the single-agent setting, the reward function is designed to optimize organ allocation across patient groups based on urgency and prioritization. The reward incorporates positive incentives for allocating organs to patients based on their assigned priority and penalizes remaining waitlist members, especially members in the high-risk group. This ensures a balance between meeting urgent needs and efficient organ utilization.
The reward for a given time step is defined as:

$$R = \sum_g a_g m_g - \alpha^T \cdot \text{waitlist}$$

Here, $\alpha \in \mathbb{R}^3$ represents penalization weights, with α_{HR} being the highest. WAITLIST includes the number of patients remaining in each group after allocation.
For the case with randomized arrivals, an additional reward function was utilized. The reward function balances incentivizing effective organ allocation while penalizing inefficiencies and unmet demand. It rewards matches based on urgency levels and group priorities, scaled to encourage allocation to higher-priority groups. Penalties are applied for failing to allocate to critical patients when medium-priority patients are ignored, running out of organs prematurely, or inefficient organ usage, while bonuses are given for improving rewards across episodes, fostering a dynamic and balanced allocation strategy.

$$R_t = 0.5 \sum_g a_g m_g + \sum_g m_g - 10\mathbb{I}(\text{waitlist}_{HR} > 0)$$
$$+10\mathbb{I}(R_t > R_{t-1})$$

In the multi-agent setting, each agent g receives:

$$R_{g,t} = a_g m_g - 2\text{waitlist}_{HR}\mathbb{I}(m_{LR} > 0 \wedge m_{MR} > 0)$$

The decentralized reward function evaluates the efficiency of organ allocation by assigning rewards to groups based on the urgency of matched patients and the priority level of each group. A penalty is applied when high-priority patients are ignored while organs are allocated to lower-priority groups, ensuring a balance between fairness and efficiency. The decentralized approach calculates rewards and penalties at the group level, promoting localized decision-making while considering global constraints like shared organ availability.

Urgency Modeling from MELD Data

To move beyond synthetic distributions, we fit real MELD scores to established probability distributions. Statistical tests (Kolmogorov–Smirnov) indicate that:

- Low-Risk MELD scores are best approximated by a Normal distribution $N(\mu = 7.6, \sigma = 1.056)$.
- Medium-Risk MELD scores fit a Beta distribution with parameters $(a = 0.916, b = 1.562)$, shifted and scaled to MELD range [10, 39.6].
- High-Risk MELD scores follow an Exponential distribution with $(\text{loc} = 40.0, \text{scale} = 3.408)$.

These distributions are mapped into urgency values $u \in [0, 10]$, ensuring that LR patients generally have lower urgencies, MR patients have mid-range urgencies, and HR patients have higher urgencies. This clinically grounded approach to modeling urgency provides a realistic foundation for allocation decisions.

Learning Framework

We apply model-free RL methods to learn policies:

Single-agent Q-learning: A single Q-table $Q(s, a)$ is learned using:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)].$$

Multi-agent Nash Q-learning: Each group g maintains its own Q-table $Q_g(s, a_g)$. After observing outcomes, Q_g is updated with:

$$Q_g(s,a_g) \leftarrow Q_g(s,a_g) + \alpha\big[r_{g,t} + \gamma \max_{a'_g} Q_g(s',a'_g) - Q_g(s,a_g)\big].$$

All agents use an ϵ-greedy exploration policy, reducing ϵ from 0.8 to 0.1 over 10,000 training episodes. As agents learn concurrently, they approximate a Nash equilibrium, balancing their strategies against the others.

Implementation

We train both single-agent and multi-agent scenarios for 10,000 episodes. Each episode simulates up to 30 days or ends early if the organ supply is depleted:

1. **Initialization**: Start with 50 organs and set $t = 0$.
2. **Daily Loop**:

- **Urgency Sampling**: Draw urgency values for a small number of patients from each group using the fitted MELD-based distributions.
- **Allocation**: Assign organs to patients whose urgency exceeds their group's priority threshold, if organs are available.
- **State Update**: Deduct allocated organs and record the new state (m_{LR}, m_{MR}, m_{HR}).
- **Learning**: Compute rewards and update Q-values for the single-agent or each agent in the multi-agent setting.

3. **Termination**: Stop if all organs are used or after 30 days.

Evaluation: We track and compare cumulative rewards per episode to assess policy convergence, stability, and equity. We also look at the organs allocated to each group at each time step against the simulated demand to determine how well the policy handles random patient arrivals.

Deep Q-Network

We also modeled the problem as a Deep Q-Network (DQN). It is a fully connected neural network designed to map state representations to action-value estimates. The input layer processes flattened state representations, ensuring compatibility with subsequent layers. Two hidden layers, each with 64 units and ReLU activations, provide the network with the capacity to learn non-linear relationships between states and actions. The output layer consists of as many nodes as there are possible actions (`max_waitlist_members`), with each node representing the Q-value of the corresponding action.

Environment Design

The environment simulates a setting where donors and recipients interact daily. Key components of the environment include:

- **Waitlist and Donor Data:** Each day where there exists a donor and a waitlist member with a blood-type match, the environment is initialized with a set of valid donors and recipients. The waitlist is a subset of the historical dataset, originally sorted by recipients' MELD scores and the number of days they have waited for a transplant, reflecting clinical priorities.
- **Sequential Allocation Process:** At each step, a donor is matched to a recipient from the waitlist. After all donors for a given day are processed, the environment advances to the next day.
- **Customizable Horizon:** The simulation operates over a user-defined time horizon, denoted as `max_allocations`, which specifies the number of allocations the simulation runs.

Environment Dynamics

The environment is parameterized by the maximum number of recipients to consider on the waitlist (`max_waitlist_members`) at each step and the maximum allocations (`max_allocations`) per episode. The state of the environment is represented by a structured encoding (high dimensional vectors) of the waitlist, including initial MELD scores, waitlist duration in days, blood-type, diagnosis, and functional status.

State and Action Spaces

$$s_t = \begin{bmatrix} \text{MELD Score}_1 & \text{Days on Waitlist}_1 & \text{Blood Type Code}_1 & \dots \\ \text{MELD Score}_2 & \text{Days on Waitlist}_2 & \text{Blood Type Code}_2 & \dots \\ \vdots & \vdots & \vdots & \vdots \\ \text{MELD Score}_n & \text{Days on Waitlist}_n & \text{Blood Type Code}_n & \dots \end{bmatrix} \quad (1)$$

The action space consists of indices corresponding to the recipients in the waitlist. Actions are constrained to valid matches based on blood-type compatibility by masking invalid blood-type matches with large negative values.

Reward Function

$$R = \log_{10}(\text{graft\_lifespan} + \text{patient\_survival\_time})$$

The reward function reflects the the sum of graft lifespan and patient survival time. This is based on the historical data. Though the donor from the historical data may not match the model's allocation, we assume similar graft and patient lifespan to remove another level of complexity.

State Transition

At each time step, the environment:

1. Processes the action, identifying the chosen recipient for the current donor.
2. Updates the set of allocated donor IDs to prevent re-allocation.
3. Advances the donor index and potentially refetches new donor and recipient data when the day's donors are exhausted.

An episode concludes when all donors across the specified days have been processed.

Memory Management

Effective memory management is crucial for training the DQN in the organ matching environment. Our implementation incorporates a replay memory buffer to enhance learning stability and efficiency by enabling experience replay and reducing temporal correlation in training data.
Replay memory is implemented as a circular buffer with a fixed capacity, allowing the environment to store and

sample transitions during training. At each time step, the agent records transitions in the form:

$$(state, action, reward, next\_state, done, info).$$

Info, here, includes information about the donor during the state in order to mask invalid actions when drawing samples from memory. Once the buffer reaches its maximum capacity, older transitions are discarded to make space for new ones, ensuring memory usage remains bounded. The memory design supports efficient storage and retrieval for large-scale simulations.
To train the DQN, we sample mini-batches of transitions from the replay memory. Sampling introduces randomness into the training process, decorrelating consecutive experiences and improving convergence. Each batch contains:

- **States:** Encoded representations of the environment at a given time step.
- **Actions:** Chosen actions by the policy network or via epsilon-greedy exploration.
- **Rewards:** Feedback signals indicating the quality of actions.
- **Next States:** Resulting states after transitions.
- **Termination flags:** Indicators of whether the episode has ended.
- **Invalid actions for the next state:** Invalid actions for next state in order to mask out invalid matches for the target policy net.

Memory Capacity and Trade-offs The size of the replay memory (`memory_capacity`) is a critical hyperparameter. A larger capacity increases the diversity of transitions stored, providing richer training data. However, it also requires more computational resources for sampling and storage. We selected a default capacity of 1000, balancing diversity and resource constraints.
The target network periodically synchronizes with the policy network every `target_update` episodes. This design stabilizes training by reducing oscillations in Q-value estimates, as the target network is updated less frequently than the policy network.

Results

Multi-Agent Markov Decision Process (MMDP)

We ran experiments under the following conditions:
Groups and Supply:
- Number of patient groups: 3 (Low, Medium, High Risk)
- Initial organ supply: 500
- Ischemic time limit: 30 (discrete steps)
- Q-learning episodes: 2000

Patient Distribution:
- Initial patients per group: Low: 2, Medium: 5, High: 1

Q-Learning Parameters:
- Exploration probability (ϵ): 0.8
- Learning rate (α): 0.3
- Discount factor (γ): 0.99

Patient Group Parameters:
- Mean new patients per group from Poisson(λ): Low: 2, Medium: 5, High: 1
- MELD distributions: Defined as above

Experiment Outcomes

The following table shows experiment outcomes for the various strategies. The average reward was calculated by running each model with the above conditions and averaging the final 200 iterations of training 10 times. As we can see, for the single-agent Q-learning, the updated reward structure produces the best average rewards.

Model	Patient Arrivals	Average Reward
Single-agent Q-learning	Steady Stream of Patients	58.86
Single-agent Q-learning	Randomized Patient Addition	9.90
Single-agent Q-learning	Randomized Patient Addition & Updated Reward Structure	62.03
Multi-agent Q-learning	Steady Stream of Patients	66.25 (Low) 68.66 (Med) 67.91 (High)
Multi-agent Q-learning	Randomized Patient Addition	58.18 (Low) 53.63 (Med) 46.15 (High)

Table 1: Average rewards per model

Convergence and Stability

We can clearly see the limitations of single-agent Q-learning by A and B. In the steady patient arrival case, we end up allocating as many organs as we can at each step setting our optimal action to 0. Effectively, we are not learning here. In the randomized arrival case, the penalty quickly takes over and our reward goes down. Once again, the optimal policy tries to immediately meet all demand. With the new reward structure (see C) our single-agent learns the reward dynamics and converges to a positive award. With Nash-Q learning and steady arrivals, although we have convergence for each of the three groups, we see that the allocation policy is not optimal D. For randomized patient arrivals, on the other hand, Nash quickly adopts an optimal policy and learns the reward dynamics even though

the convergence reward is lower E. Finally, we can see how steady-arrivals and single-agent allocations lead to 'perfect' allocations, while the multi-agent Nash model shows competition between groups D and A.

Deep Q-learning

Training:

- Number of episodes: 200
- Dates selected: at random but limited to start dates within the first five months of 2020

Test:

- Number episodes to average over: 10
- Maximum allocations: 500
- Dates selected: November 1st-November 29th 2020

Model parameters:

- Batch size: 128
- Learning rate (α): 0.001
- Epsilon start (ϵ_s): 1
- Epsilon end (ϵ_e): 0.01
- Memory capacity: 1000
- Target policy update interval: 200

The following numbers should be compared against the reward of the original dataset (i.e. the sum of patient and graft lifespans for all transplants conducted over the same window), which was $1,656$. These include the most successful experiments run.

Waitlist size	**Organs**	**Reward**
250	100	1,566
250	150	1,564
500	100	1,561
500	150	1,555

Table 2: Rewards per experiment based on episodes, waitlist size, and number of maximum allocations. Includes historical comparisons.

Increasing the number of allocations per episode did not lead to an increase in performance. However, lowering the number of waitlist members did improve model performance, which intuitively makes sense given the smaller decision space. The differences were not large between experiments and rewards plateaued at episode 150. The DQN performed very close to the historical averages in terms of patient and transplant lifespan. This is likely due to the limitations of the waitlist window. The DQN was only given the (`max_waitlist_members`) from the historical dataset, ordered by MELD score. Therefore, its ability to maximize patient lifespan was limited to only the most urgent cases. We see this in the distributions in Figure 16, 17, 18, and 19, where the historical dataset had a much wider distribution over MELD scores. The age distributions did not vary greatly between the model and the historical set (Figure 24, 25, 26, and 19) with the model having a slightly higher shift towards younger recipients. The DQN model did select a higher concentration of patients with better functional status (Figure 20, 21, 22, and 23) (lower scores indicate they could more easily conduct daily tasks without assistance). This makes sense considering it optimizes for lifespan and those with better mobility will likely experience a longer lifespan.
Future work might explore whether changing the ϵ or γ parameters can these dynamics in a positive way. DQN inherently requires a lot of hyper-parameters, however we had limited GPUs for experimentation, so we anticipate there is room for improvement within our framework.

Discussion

By reframing organ allocation as a multi-agent game and incorporating real MELD-based urgency modeling, we show that Nash Q-learning can yield more equitable outcomes than single-agent methods. While urgency remains a critical factor—high-risk patients still gain an advantage—competition from other risk groups mitigates extreme imbalances. The resulting equilibrium better aligns with fairness objectives, suggesting that decentralized decision-making could enhance existing allocation frameworks.

The Q-Network performs just under the historical dataset in 2020, without regard to equitable outcomes or geography. This is likely due the fact that the features were based primarily on MELD scores, days of waitlist, functional status, and blood-type match rather than robust indicators of health and longevity, which are less salient indicators of future health outcomes. It's possible that adding features like age, and blood levels might improve life expectancy predictions and improve overall reward.

Future research could incorporate more patient attributes (e.g., blood type, donor-recipient compatibility) with the MMDP approach or explore different equilibrium concepts. Introducing explicit fairness or policy constraints into the reward functions may further guide the system toward ethically and socially desirable outcomes. These extensions have the potential to inform practical policy adjustments, balancing survival urgency and equity under severe resource constraints. Future work may also consider incorporating equitable outcomes into the DQN approach, which may result in both outperforming historical outcomes and ensuring equity.

References

[1] United Network for Organ Sharing. How we match organs.

[2] United Network for Organ Sharing. Medical urgency in liver allocation: MELD and PELD scores.

[3] Freeman, R. B., Wiesner, R. H., Roberts, J. P., McDiarmid, S., Dykstra, D. M., & Merion, R. M. (2004). Improving liver allocation: MELD and PELD. American Journal of Transplantation, 4:114–131.

[4] Kochenderfer, M. J., Wheeler, T. A., & Wray, K. H. (2022). Algorithms for decision making. MIT Press.

[5] Fan, W., Zong, Y., & Kumar, S. (2020). Optimal Treatment of Chronic Kidney Disease with Uncertainty in Obtaining a Transplantable Kidney: An MDP-based Approach. Annals of Operations Research, 316:269-302.

[6] Sandıkçı, B., Maillart, L. M., Schaefer, A. J., Alagoz, O., & Roberts, M. S. (2008). Estimating the Patient's Price of Privacy inLiver Transplantation. Operations Research, 56(6), 1393-1410.

[7] M. Massey, Dynamic Prioritization for Organ Matching with Model-Free Reinforcement Learning. CS238, Fall 2023.

[8] Wiesner, R., Edwards, E., Freeman, R., Harper, A., Kim, R., Kamath, P., & Kremers, W. (2003). Model for end-stage liver disease (MELD) and allocation of donor livers. Gastroenterology, 124(1), 91–96.

[9] Berger, V. W., & Zhou, Y. (2014). Kolmogorov–Smirnov test: Overview. Wiley StatsRef: Statistics Reference Online.

[10] Hosseini, H., Hoey, J., & Cohen, R. (2014). A coordinated MDP approach to multi-agent planning for resource allocation, with applications to healthcare.

[11] Department of Health and Human Services, Health Resources and Services Administration, Healthcare Systems Bureau, Division of Transplantation, United Network for Organ Sharing, & University Renal Research and Education Association. (2024). 2024 annual report of the U.S. Organ Procurement and Transplantation Network and the Scientific Registry of Transplant Recipients: Transplant data 1988–2024. Rockville, MD

Acknowledgement: This work was supported in part by Health Resources and Services Administration contract HHSH250-2019-00001C. The content is the responsibility of the authors alone and does not necessarily reflect the views or policies of the Department of Health and Human Services, nor does mention of trade names, commercial products, or organizations imply endorsement by the U.S. Government.

[12] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. nature, 518(7540), 529-533.

Appendix

A: Single Agent Q-Learing with Steady Patient Arrivals

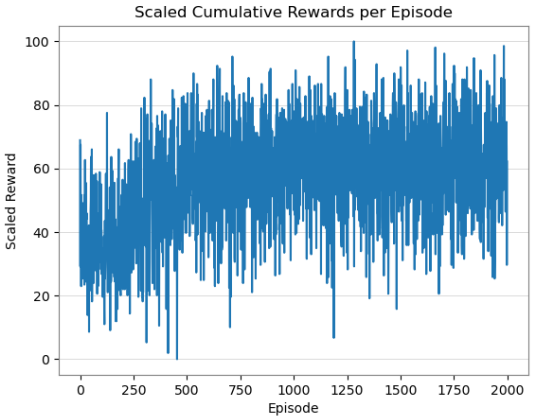

Figure 1: Cumulative rewards for steady arrivals in single agent Q-Learning

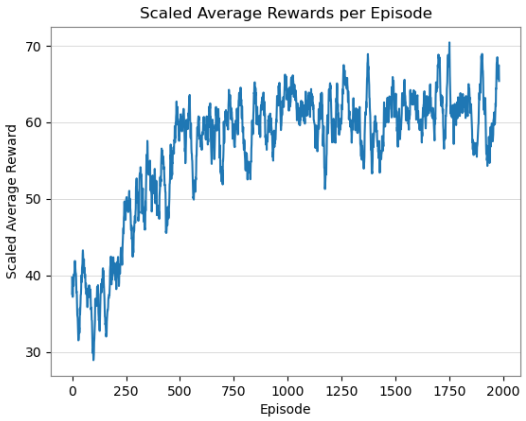

Figure 2: Smoothed rewards for steady arrivals in single agent Q-Learning

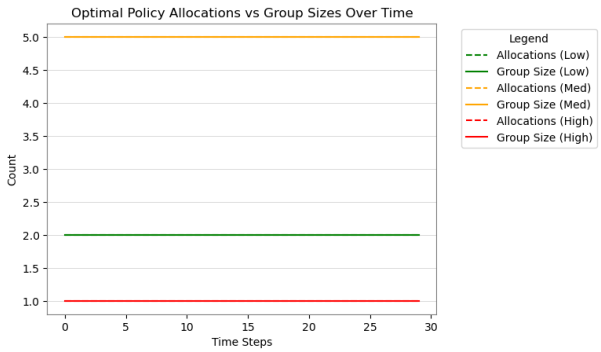

Figure 3: Optimal allocation policy with simulated demand

B: Single Agent Q-Learing with Randomized Patient Arrivals

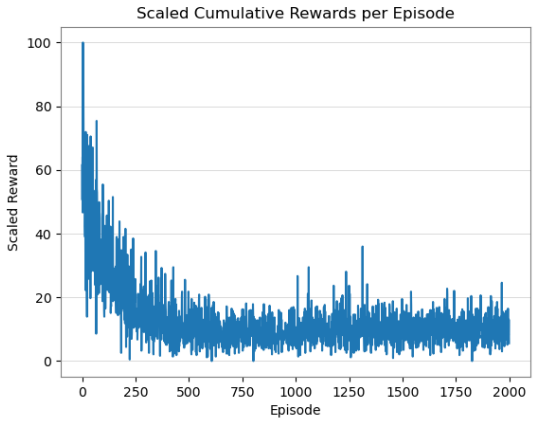

Figure 4: Cumulative rewards for randomized arrivals in single agent Q-Learning

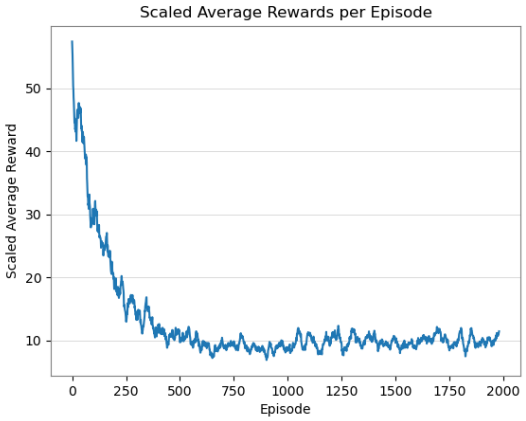

Figure 5: Smoothed rewards for randomized arrivals in single agent Q-Learning

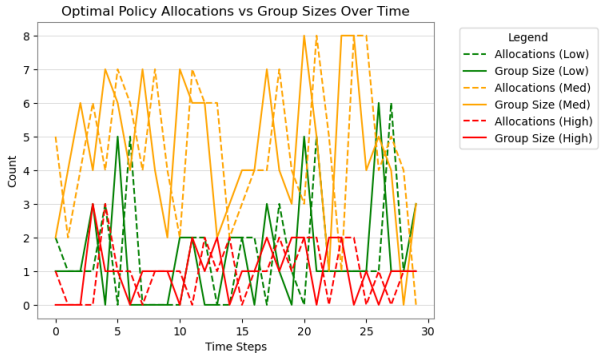

Figure 6: Optimal allocation policy with simulated demand

C: Single Agent Q-Learing with Randomized Patient Arrivals and New Reward Function

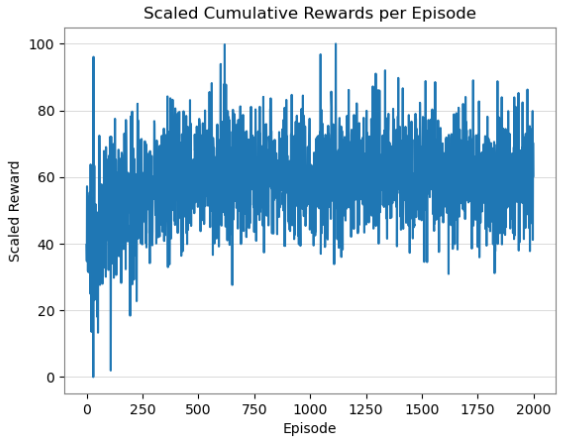

Figure 7: Cumulative rewards for randomized arrivals in single agent Q-Learning

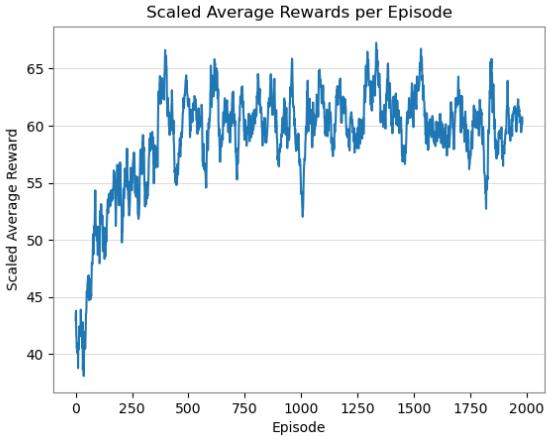

Figure 8: Smoothed rewards for randomized arrivals in single agent Q-Learning

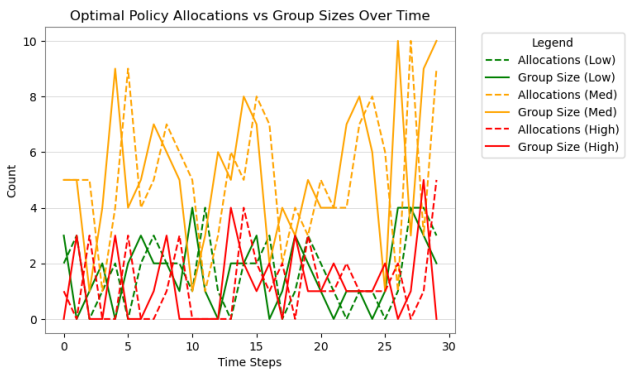

Figure 9: Optimal allocation policy with simulated demand

D: Multi-Agent Nash Q-Learing with Steady Patient Arrivals

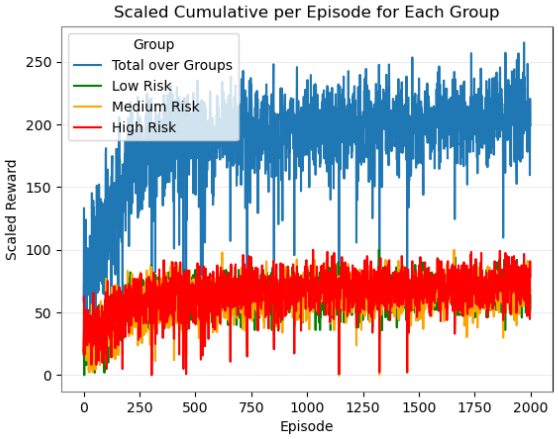

Figure 10: Cumulative rewards for steady arrivals in multi-agent Nash Q-Learning

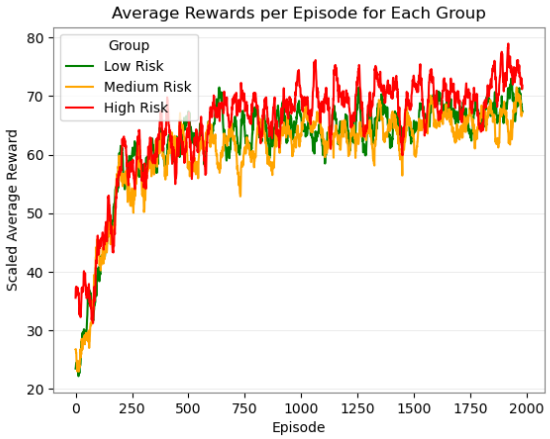

Figure 11: Smoothed rewards for steady arrivals in multi-agent Nash Q-Learning

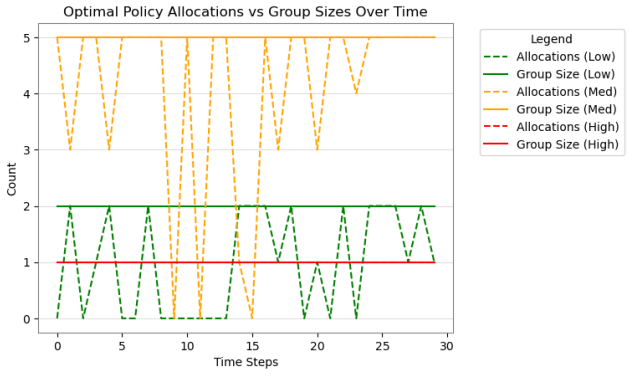

Figure 12: Optimal allocation policy with simulated demand

E: Multi-Agent Nash Q-Learing with Randomized Patient Arrivals

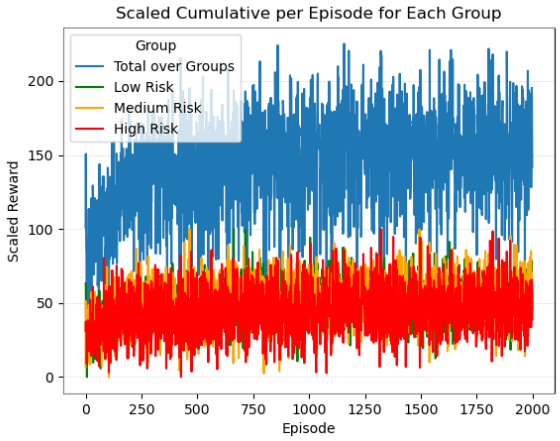

Figure 13: Cumulative rewards for random arrivals in multi-agent Nash Q-Learning

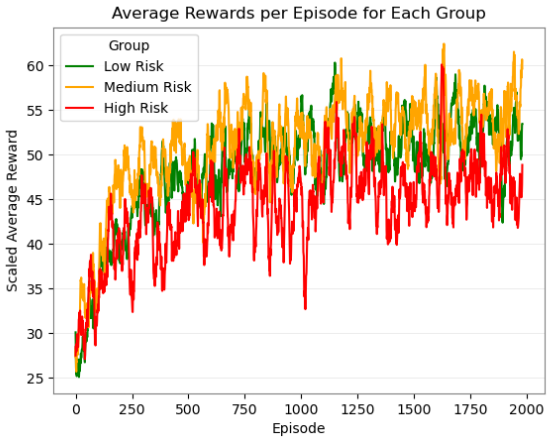

Figure 14: Smoothed rewards for random arrivals in multi-agent Nash Q-Learning

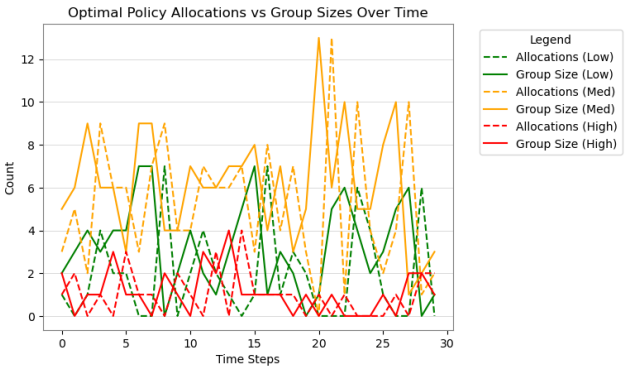

Figure 15: Optimal allocation policy with simulated demand

F: Deep Q-learning

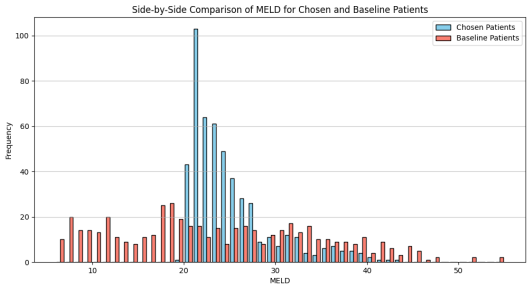

Figure 16: DQN test allocation MELD distributions for waitlist size 250, maximum allocations 100

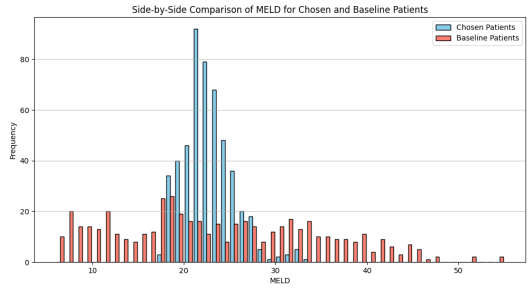

Figure 17: DQN test allocation MELD distributions for waitlist size 500, maximum allocations 100

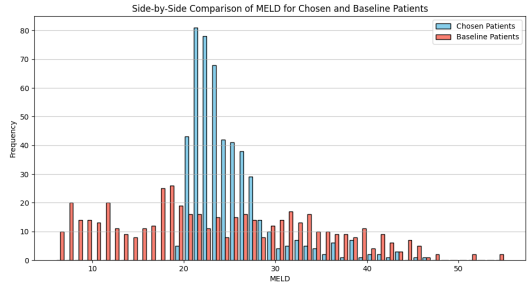

Figure 18: DQN test allocation MELD distributions for waitlist size 250, maximum allocations 150

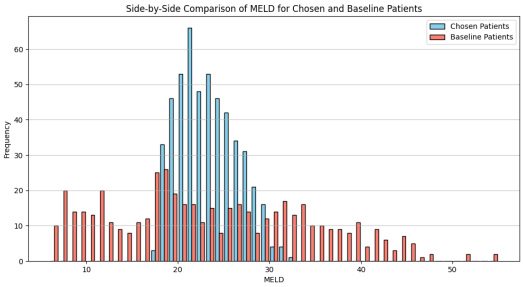

Figure 19: DQN test allocation MELD distributions for waitlist size 500, maximum allocations 150

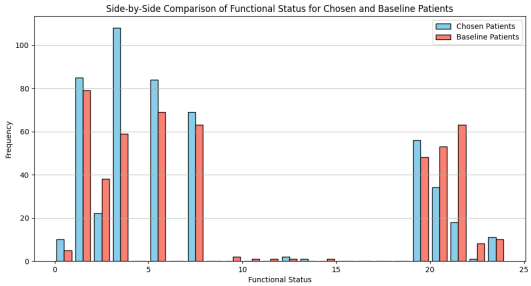

Figure 23: DQN test allocation functional status distributions for waitlist size 500, maximum allocations 150

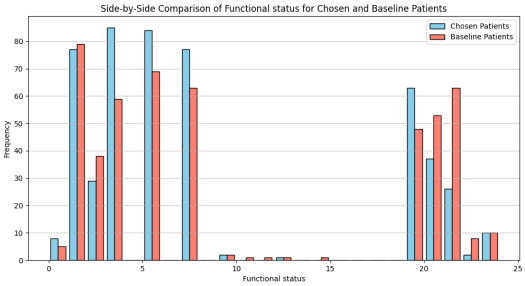

Figure 20: DQN test allocation functional status distributions for waitlist size 250, maximum allocations 100

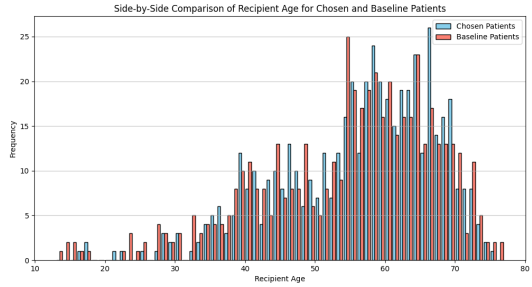

Figure 24: DQN test allocation age distributions for waitlist size 250, maximum allocations 100

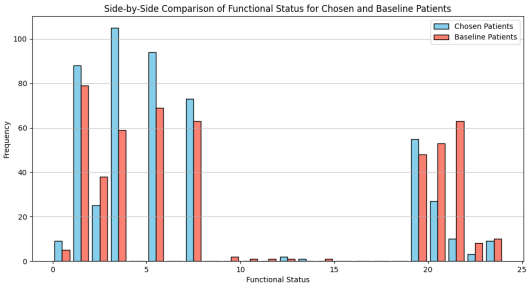

Figure 21: DQN test allocation functional status distributions for waitlist size 500, maximum allocations 100

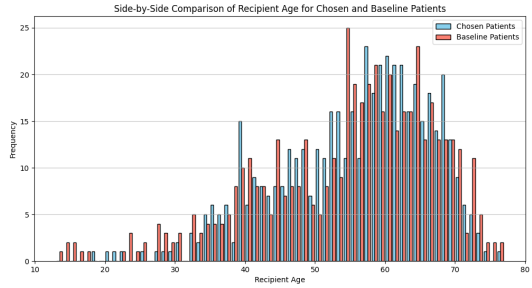

Figure 25: DQN test allocation age distributions for waitlist size 250, maximum allocations 150

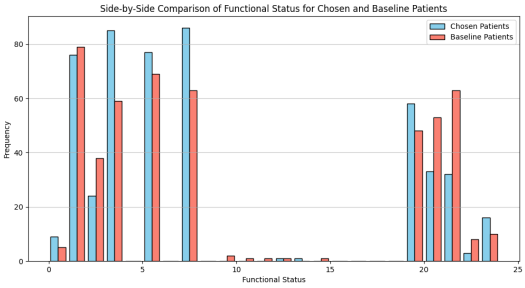

Figure 22: DQN test allocation functional status distributions for waitlist size 250, maximum allocations 150

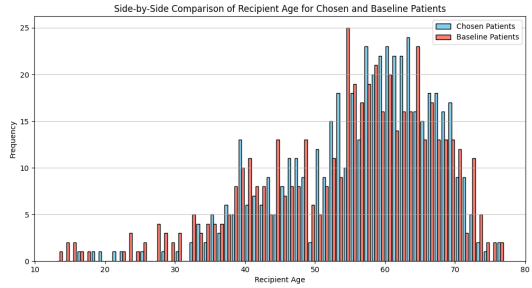

Figure 26: DQN test allocation age distributions for waitlist size 500, maximum allocations 100

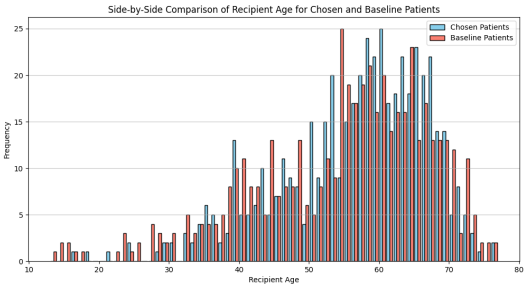

Figure 27: DQN test allocation age distributions for waitlist size 500, maximum allocations 150