Extended Abstract

Motivation When kidney transplant patients have a family or friend willing to donate with whom they're incompatible, Kidney Paired Donation (KPD) programs seek to match patient-donor pairs with one another. Matching pairs that arrive and depart over time introduces a complex trade-off between matching as soon as possible and waiting for potentially better future matches. Traditional approaches either simplify the matching decision criteria or optimize offline; a recent Approximate Dynamic Programming (ADP) algorithm improves on this but still models the policy with approximated basis functions, rather than leveraging the complexity of neural networks. This paper sharpens the problem by modeling KPD as a granular discrete-time Markov Decision Process on a time-bound compatibility graph and asks: can we leverage a Deep Q Network algorithm with a Graph Neural Network model to learn a policy that can maximize match volume and optimize for such complex (or even counteractive) objectives as fairness?

Method We discretize time into fine-grained steps and represent the evolving registry as a graph whose nodes are patient–donor pairs (with medical and demographic attributes) and whose edges indicate feasible transplants. We implement a Double Deep Q-Network (DDQN) whose Q-function is parameterized by a Graph Neural Network (GNN), allowing the policy to reason over complex relational features. At each timestep, the agent selects at most one edge to realize as a match. We first compare the performance of DDQN to that of the ADP algorithm on the same dataset. Then three reward functions are explored – pure match-count efficiency, a group-fairness penalty based on deviation from a target demographic distribution, and a weighted combination of the two – to explore the trade-off between efficiency and fairness.

Implementation Using a synthetically generated KPD dataset calibrated to real UNOS and National Kidney Registry statistics, the first experiments compare DDQN to the state-of-the-art ADP algorithm over a range of explored hyperparameters. Each simulation episode spans 50 or 100 discretized timesteps, though we experimented with up to 300 timesteps to probe stability. The GNN encoder uses two message-passing layers with hidden dimension 256 and edge embedding size 32. The DDQN agent employs a replay buffer of size 10 000, a batch size of 256, a learning rate of 1e-3, discount factor $\gamma = 0.99$, and an ϵ-greedy schedule decaying from 1.0 to 0.05 over 50 episodes. Every 100 timesteps we update a target network. With these settings, we compared the performance (match rate over episodes) of the DDQN algorithm to that of the ADP algorithm sampling 50 graph states per iteration. Then to experiment with fairness, we used the same efficiency reward (r_e), a pure feature-distribution reward (r_f), and a mixed reward (r_m with $w_e = w_f = 0.5$) over 20 timesteps per episode and 50 episodes and compared the match rate, reward, and loss for each reward function.

Results Against our simulated ADP baseline of You and Vossen (2024) which samples 50 graph states per iteration and solves a reduced LP with basis functions—the DDQN policy achieves an average match rate of 0.20 (≈ 9.5 matches per 50-step episode), compared to ADP's 0.18 (≈ 8.3 matches). Learning curves show DDQN's per-episode match rate fluctuating within ± 0.02 of its mean, indicating limited policy learning; loss remains roughly constant at 0.11 MSE. In fairness experiments, match rates rose slightly to 0.12 and 0.122 respectively versus 0.044 under pure efficiency. However, the MSE loss under r_f and r_m climbed to 0.93 and 1.16, suggesting unstable Q-value estimation when balancing more complex reward terms. Reward traces for r_f and r_m(Figure 5) exhibit high volatility—up to $\pm 30\%$ of the mean, underscoring sparse feedback for fairness objectives.

Discussion & Conclusion The competitiveness of an untrained DDQN against ADP suggests deep RL holds promise for dynamic KPD, but failure of the model to learn reveals challenges: sparse and imbalanced rewards, large state–action spaces, and computational burdens for dataset and model complexity. The paper highlights the need for richer experience reuse, refined reward shaping, and perhaps curriculum learning or hierarchical modeling to spur convergence.

We experimented with using a DDQN algorithm with a GNN model to solve the complex MDP of KPD matching. We found that the baseline performance of the algorithm is at level with the current KPD matching ADP algorithm, but the agent failed to learn under the conditions we set. We also experimented with group fairness-informed reward functions, and similarly found comparable baseline match rates across reward functions but no learning, rendering our analysis of the efficiency-fairness tradeoff inconclusive. While the proposed framework opens avenues for balancing efficiency and group fairness, realizing its potential will require both algorithmic innovations to accelerate learning and careful policy-maker guidance to set fair but practical target distributions.

Dynamic Kidney Exchange with Deep Q Networks

Odelia Lorch
Department of Computer Science
Stanford University
olorch@stanford.edu

Abstract

This paper seeks to address the problem of dynamic matching for Kidney Paired Donation (KPD) by leveraging deep reinforcement learning to train a policy that decides which matches to make and when. We characterize the problem as a Markov Decision Problem (MDP), modeling the state of the KPD market as a time-bound graph whose nodes are patient-donor pairs and edges are feasible matches, and the action space as the match made, if any, at each discretized timestep. We then apply a Double Deep Q Network (DDQN) algorithm to train a Graph Neural Network (GNN) policy to solve this MDP. We first train the model with a reward function that maximizes matches made over time ("efficiency"), and compare the performance of our DDQN to that of the recent successful Approximate Dynamic Programming (ADP) algorithm devised by You and Vossen (2024). We then train the model with a reward function that maximizes group fairness across a given feature of the patient-donor pair, as well as a reward function that combines the efficiency and fairness objectives, and compare the performance of our algorithm to explore the tradeoff between efficiency and fairness in KPD matching. Our findings, while demonstrating that DDQN performs at least as well as ADP, remain inconclusive, which leads us to understand the complexity of this MDP and conjecture ways to improve our model.

1 Introduction

When a patient needing an organ donation has a relative or friend who is willing to donate, but the donor's organ is not the right match for the patient, the patient-donor pair often seek to undergo a paired exchange. About 1 in 5 living kidney donations in 2021 were paired donation exchanges, while the vast majority of the rest of living donations are direct, and only a small percentage altruistic (a donor without a patient) Garg et al. (2024). There is still much work to be done in improving paired organ exchanges, from better understanding compatibility and success probabilities to optimizing algorithms to maximize metrics such as matches over time or fairness of matches to various groups of patients and donors. This paper is concerned with the latter, optimizing matching algorithms.
The pool of patient-donor pairs that are waiting to be matched can be modeled as a graph, whose nodes are each one such patient-donor pair and edges are feasible matches, possibly weighted by their level of suitability. Each node is described by the various necessary medical information about the donor and patient, as well as relevant demographic and socioeconomic information, for the purpose of our fairness objective. This graph is modeled over time, pairs entering the graph as they arrive and leaving the graph when either a match is made or a patient becomes no longer eligible. Furthermore, the state of each node (i.e. illness development, eligibility) is updated over time. The question of whom to match and when is a Markov decision problem, yet one that is particularly combinatorially complex, and whose transition probabilities are uncertain. The question this paper explores is: how might we leverage reinforcement learning to tackle this challenging MDP and optimize a two-fold objective, one of maximizing the number of matches and the other of ensuring group fairness across matches?

2 Related Work

A longstanding research question in kidney paired donation (KPD) has been how to optimize the number of matches made over time when patient-donor pairs arrive dynamically. If a certain match is made at time t, how can we be sure it does not preclude a higher-weighted match at time $t + 1$? Prior research typically uses the graph model described in Section 1 to represent the KPD "market." Because KPD programs explicitly identify potential matches between participants, the state of the "market" described by this graph model is well-defined. Therefore the matching problem is generally approached as a complex Markov decision problem (MDP). In their seminal paper addressing this problem, Akbarpour et al. (2020) highlighted time, as in the choice of when to match, as a "first-order concern" in this optimization problem, and rather than address the MDP head-on, demonstrate that a simple delayed-matching algorithm comes relatively close to optimal performance.

Some research has been done in leveraging reinforcement learning to optimize the success of the matches once they are made, as in the work of Deshpande (2024) and reviewed by Olawade et al. (2025). The question of using reinforcement learning to solve the MDP of matching patient-donor nodes, however, has yet to be addressed.

Much work has been done to address the question of fairness in matching problems, both bipartite and nonbipartite. Sankar et al. (2021) analyze classified maximum matching (CMM) with a "group fairness" concept that assigns maximal quotas to each subset within a collection of laminar subsets of the set of "agents" that are feasible matches with some "platform," and proves a good competitive ratio for offline and online random input arrival models with this constraint. Ma et al. (2023) and Esmaeili et al. (2023) both incorporate Rawlsian, or maximin fairness, in offline and online bipartite matching, by seeking to maximize the minimum expected matching rate over all individuals on one side of the graph. Hosseini et al. (2023) presents deterministic algorithms and approximation bounds for several concepts of fairness for a partition of one side of the graph into classes: class envy-freeness, class proportionality, and class maximin share fairness. Finally, the recent paper Castera et al. (2024) analyzes added loss from incorporating a good survey of fairness constraints in *offline* bipartite matching: Shapley fairness, leximin fairness, as well as the group fairness notions of demographic parity and equal opportunity.

For nonbipartite dynamic kidney exchange, Dickerson and Sandholm (2014) introduced two notions fairness: (1) prioritizing matches that are more likely to succeed by weighting the utility of matches by their probability of success and (2) a simple version of group fairness in which patients either do or do not belong to the "marginalized group," and if they do, their match utilities are weighted by some constant factor across all members of the group (Gao (2019) later discusses the tradeoff between these two types of fairness in dynamic matching). More recently, St-Arnaud (2021) proposed a framework for individual fairness in dynamic kidney matching, which minimizes the L_p average of the discrepancy of each individual patient's probability of receiving a match from the average (compared with the algorithmic fairness definition of individual fairness, this assumes that all patients are equally worthy of a match), and argues its advantages over Rawlsian fairness, Nash's proportional fairness, Shapley value schemes, and group fairness for "hard-to-match" patients.

3 Method

We apply a Double Deep Q Network (DDQN) algorithm to learn a policy for dynamic matching in the setting of kidney paired donations (KPD). We optimize for the objectives of "efficiency," i.e. maximizing the total number of matches over time, "fairness," namely satisfying a certain fairness metric, and the combined objective of both efficiency and fairness. We compare performance to understand the severity of the fairness-efficiency tradeoff in dynamic KPD matching.

3.1 Dataset

Because UNOS, National Kidney Registry, and similar real KPD datasets are only available via collaboration requests, for the timeline of this project we used synthetic data. Publicly-available synthetic KPD datasets exist (Dickerson et al. (2013)) and have been used in various research, but we found that these datasets were lacking the resolution and features necessary for this project. Therefore we generated a new synthetic dataset based on statistical data about KPD and distributions of various features for KPD in recent years (see Appendix A for a review of these statistical distributions).

Table 1: Performance Comparison

	Hyperparameters	Default
DDQN Agent	`learning-rate`	1e-3
	`gamma` (discount factor)	0.99
	`buffer-cap` (capacity of replay buffer)	10000
	`batch-size`	64
	`target-update` (frequency)	100 timesteps
GNN Model	`hidden-dim`	64
	`edge-dim`	32
Training Loop	`num-episodes`	50
	`eps-start` (starting ϵ value)	1.0
	`eps-end` (ending ϵ value)	0.05
	`eps-decay`	0.995
Reward Function	`reward-type`	`simple` (r_e)
	`feature-name` (for r_f)	`patient-gender`
	`desired-dist` (p_i for feature i)	Male:0.5,Female:0.5
	`reward-e` (w_e for r_m)	0.5
	`reward-f` (w_f for r_m)	0.5

These data came from National Kidney Registry (2024), Holscher et al. (2018), Flechner et al. (2018), Stepkowski et al. (2019), and Leeser et al. (2020).
The dataset is set up as follows. The "market" is modeled as a graph. The nodes of this graph are donor-patient pairs, with non-time-bound features `patient-gender`, `patient-race`, `patient-age`, `patient-blood` type, `donor-gender`, `donor-race`, `donor-age`, `donor-blood` type, and `incompat-reason` reason for incompatibility between the donor and patient; and time-bound features `patient-health` ("good," "fair," or "poor"), `patient-cpra` cPRA score, donor `antibodies`, and donor `antigens`. The edges of this graph represent feasible matches.
At each timestep, the environment naturally can cause a new donor-patient pair to enter the "market," a donor-patient pair to leave the "market," and/or the time-bound features of any set of nodes to change. We set the number of timesteps per year to 8000; the more timesteps per year, the closer this model gets to approaching continuous time. Because each of these changes is atomic, in a continuous-time MDP we can expect no two events to happen simultaneously. This choice of a large number of timesteps per year allows us to approximate this phenomenon. In particular, according to the National Kidney Registry (2024), the average number of patient-donor pair arrivals per day is 10, so with $8000/365 > 10$ timesteps per day, the expected number of arrivals per timestep is less than 1. We introduced randomness to the data simulation code such that it will generate a distinct dataset each time. This allows us to train and experiment "different" datasets.

3.2 RL Algorithm

In Section 3.1 we described the state space of our model. The action space is the choice, at each timestep, of which pair of nodes to match (which feasible-match edge to realize), if any. Transition probabilities are obscure in this MDP, so deep RL is a promising approach.
The available real datasets for KPD are small, so rather than simulate large datasets when such do not exist in reality, we decided to leverage off-policy algorithms. The true action space is partly combinatorial (edge choice) and partly continuous (timing), which poses a challenge for choosing an RL model. This is why we chose to attempt discretizing time at relatively high resolution so that the action space would be purely combinatorial. This allows us to apply a Double Deep Q-Network (DDQN) algorithm to this model, using a Graph Neural Network (GNN) for the policy.
We applied this DDQN with each trajectory defined as a set number of timesteps in which the state of the graph changes in an online manner according to both the dataset and the actions taken (matches made) by the algorithm. The tunable hyperparameters of our algorithm are enumerated in Table 1.

3.3 RL Objectives

We experiment with three different reward functions to optimize for efficiency (match rate), fairness, and a combination of the two. Because we are using a DDQN, the loss function is the mean-squared-error between Q-network's Q values and the target Q values, discounted by some $\gamma \in (0, 1)$.

Our first reward function maximizes the total number of matches over time. For any timestep t, the ***efficiency*** function is defined as

$$r_e(t) = \begin{cases} 1 & \text{if a match is made at time } t \\ 0 & \text{otherwise} \end{cases}$$

Once we have maximized the total number of matches to the best of our ability, we introduce a group fairness objective. For any node feature, for example `patient-gender`, `donor-race`, or `patient-health`, let $S = \{1, ..., n\}$ be the set of types for that feature. Given a desired match distribution vector $p \in [0, 1]^n$ over these types (chosen by the policy-maker implementing this algorithm), we define the ***group fairness*** reward function at timestep t to be the mean-squared-error between p and the per-group match rates at time t:

$$r_f(t) = \frac{1}{n} \sum_{i \in S} \left(p_i - \frac{\text{\# matches of type } i \text{ by time } t}{\text{total \# of matches by time } t} \right)^2$$

Finally, we define ***mixed fairness*** r_m as a linear combination of r_e and r_f:

$$r_m(t) = w_e \cdot r_e(t) + w_f \cdot r_f(t)$$

where $w_e, w_f \in [0, 1]$ are weights chosen by the policy-maker such that $w_e + w_f = 1$.

4 Experimental Setup

We first sought to understand the effectiveness of our DDQN algorithm with respect to the standard objective of maximizing the rate of matches per patient-donor arrivals. An inherent challenge in our choice to simulate the dataset is that there are so many extraneous factors that influence KPD matching, such as physical distance between pairs, human error / cooperation of donation centers, nondescript medical factors, and so on. For this reason, we found it inappropriate to compare our findings, namely the match rates achieved by our algorithm, to the match rates that occur in reality. Instead, we chose to compare our DDQN algorithm's performance to that of a current algorithm that has gained traction, the Approximate Dynamic Programming (ADP) approach of You and Vossen You and Vossen (2024). See Appendix B for a description of the ADP algorithm. We ran both algorithms on the same dataset parameters and examined their match rates across episodes (for DDQN) and samples (for ADP). For this comparison, DDQN was run with reward function r_e to maximize efficiency.

We then sought to explore the potential of DDQN to optimize for group fairness. We ran the algorithm using the efficiency reward function r_e, then the fairness reward function r_f over the `patient-gender` feature, then the mixed reward function r_m, with its r_f component over the same `patient-gender` feature. We then compared the match rates, reward, and loss of the DDQN algorithm across these functions.

5 Results

For our first experiment, we ran ADP over trajectories of 20 timesteps per sample for 50 samples, and we ran DDQN over trajectories of 20 timesteps per episode for 50 episodes, with all other hyperparameters set to default (as outlined in Table 1). We found the match rates per sample/episode to be quite variant and unstable, which is to be expected when the arrival rate is $\lambda \approx 0.46$ patient-donor pairs per timestep, leaving approximately 10 total node arrivals per trajectory.

We then increased the number of timesteps per trajectory, and noticed that at around 300 timesteps per trajectory the algorithm's computing time became quite slow, though from any number from 100 timesteps per trajectory and above seemed to yield the same variability in outcomes across samples/episodes. Increasing the number of timesteps (to 50, then 100, 200, 300, 400) added stability, but we found that both ADP and DDQN produced results that hovered around the same average value across samples/episodes. This implied that our DDQN was not learning.

This led us to try altering the `num-episodes` (from 50 to 100 and 150), `learning-rate` (1e-3, 1e-4, and 1e-5), the `batch-size` from 64 to 256, and the GNN Model's `hidden-dim` from 64 to

Table 2: ADP and DDQN Performance Comparison

Algorithm	Match rate	Matches
ADP	0.18	8.30
DDQN	0.20	9.45

Table 3: DDQN Reward Functions Performance Comparison

Reward function	Match rate	Reward	Loss
r_e	0.044	0.33	0.11
r_f	0.120	5.86	0.93
r_m	0.122	9.32	1.16

256 as well. After several trials of tuning these parameters, our most salient results are as shown in table 2 and figures 1, 2, and 3, under `timesteps` 100, `num-episodes` 50, `learning-rate` 1e-3, `batch-size` 256, and `hidden-dim` 256.
Our next experiment involved running DDQN with three different reward functions: `simple` (r_e), `feature-distribution` (r_f), and `mixed` (r_m). We ran all three on 20 timesteps per episode for 50 episodes, with `learning-rate` 1e-3 and `batch-size` 64. For the fairness reward function r_f we ran the experiment over the distribution of `patient-gender` with a desired distribution of (0.5 Male, 0.5 Female). We ran the mixed reward function with r_f set with the same parameters. The results can be seen in table 3 and figures 4, 5, and 6.

5.1 Quantitative Evaluation

As detailed in table 2, the average (across 50 samples/episodes) match rates of ADP and DDQN are 0.18 and 0.20, respectively. Though the difference in rates is minor, we still note that DDQN seems to outperform ADP, even without the model learning and improving across episodes.
In table 3 we see a comparison of the average (across 50 episodes) performance of DDQN with three different reward functions. We note that, surprisingly, the match rate is higher when the algorithm uses a reward function that incorporates our fairness metric (r_f) and highest when the reward is mixed (r_m), partly optimizing for efficiency (r_e) and partly optimizing for fairness (r_f). This goes against our expectations, as we would imagine that a reward function that purely optimizes for number of matches would yield the highest rate, that a reward function that purely optimizes for fairness would detract from the match rate, and a mixed reward function would demonstrate the tradeoff between the two. Similarly, the average reward value for r_e is 0.33, much lower than the 5.86 of r_f and the 9.32 of r_m. What does align slightly better with our hypothesis, though, is the average loss values: r_e yields the lowest, at 0.11, while r_f and r_m are much closer to 1, at 0.93 and 0.16, respectively.

5.2 Qualitative Analysis

Figure 1 shows the match rate per episode for DDQN under these parameters in orange, and the match rate per sample for ADP with 100 timesteps per sample over 50 samples in pink. Figure 2 shows the raw number of matches (ignoring the varying number of patient-donor pairs per episode) for DDQN and ADP with these same parameters. Finally, figure 3 depicts the reward per episode for DDQN under these parameters. As one can see, the DDQN model still did not learn or improve, and the difference in average performance between ADP and DDQN is minor. Still, we can see that even without learning, the DDQN policy performs competitively with the ADP algorithm, and its average match rate is even slightly higher.
The reason DDQN is not learning could be manifold. We consider the possibility that more extensive experimenting with tuning hyperparameters, including more subtle ones such as slowing the `epsilon-decay` for the training loop so that the agent continues exploring and gathering data for longer, tuning various dimensions of the GNN model, or increasing the replay buffer's capacity to hold more experiences, may reveal conditions under which the DDQN agent is able to learn and improve the policy. We also consider that training each episode on a new trajectory might not give the algorithm enough of an opportunity to specialize its learning. Therefore using the same trajectory for several episodes may also improve the agent's learning.
Figure 4 shows the match rate per episode for the `simple` reward (pink), `feature-distribution` reward (orange), and `mixed` reward (purple). Similarly to our first experiment, the DDQN algo-

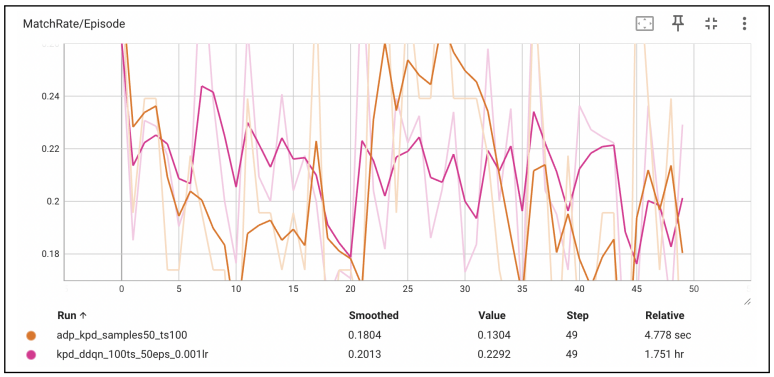

Figure 1: Match rate per episode/sample for DDQN and ADP with 100 timesteps over 50 episodes/samples.

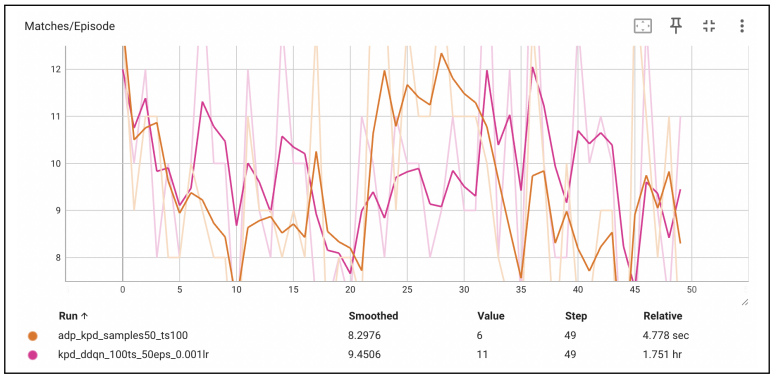

Figure 2: Number of matches per episode/sample for DDQN and ADP with 100 timesteps over 50 episodes/samples.

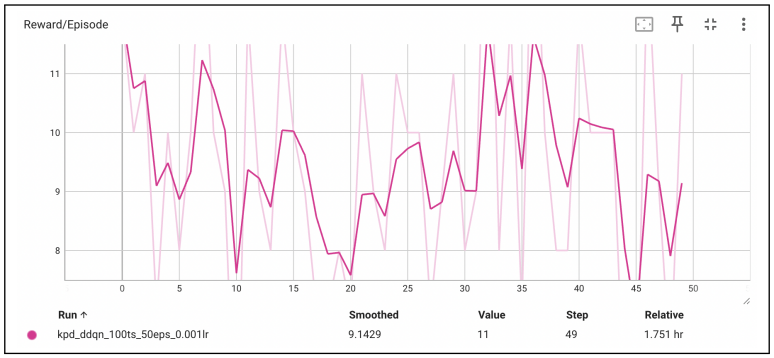

Figure 3: Reward per episode for DDQN with 100 timesteps over 50 episodes.

rithm does not appear to be learning, so the match rates hover around the same average values across episodes. Figure 5 shows the reward per episode for each reward function, and we can see that the `simple` reward maintains a low stable (if not somewhat decreasing) value, whereas the `feature-distribution` and `mixed` rewards vary widely around higher average values. Finally, figure 6 depicts the loss per episode for each reward function, and it is noteworthy that the loss for `feature-distribution` and `mixed` rewards is increasing, rather than decreasing.

The lack of growth across match rates and reward suggest the DDQN algorithm is failing to learn. The increasing loss values for the algorithm under fairness-informed reward functions (r_f and r_m) is another surprising result, that indicates that the policy's predictions are getting worse over time. We enumerated above several possible causes of this phenomenon. In our experimenting on reward functions, we also note that the reward functions may provide reward that is too scarce or infrequent, as well as uneven in scale between r_e and r_f, which might be causing the unexpected disparities between r_e and the two fairness-informed reward functions. This suggests the need to re-evaluate the reward functions, along with increasing the data available to the agent to learn.

One final consideration is that we experimented with group fairness for only one feature, `patient-gender`. It is possible that the small number of groups or the choice of uniform distribution for this feature made it more challenging to achieve productive results and contributed to our counterintuitive results. In future research, we hope to be able to experiment with group fairness for each of the patient-donor pair features with a variety of target distributions. Furthermore, we are curious to experiment with incorporating multi-group fairness into our reward function to achieve fairness for multiple features at once.

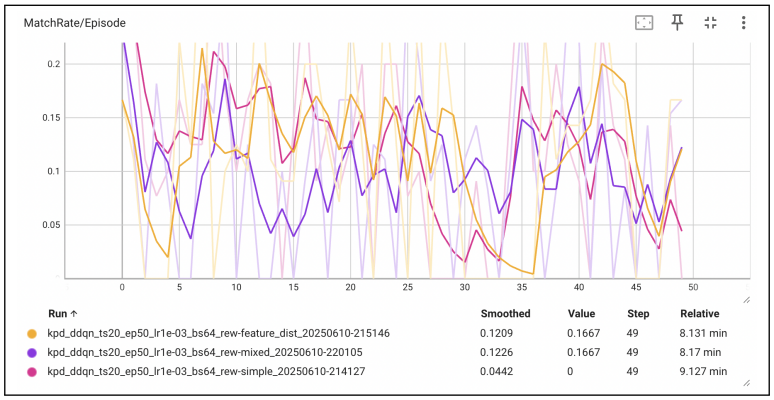

Figure 4: Match rate per episode for DDQN with 20 timesteps over 50 episodes under reward functions `simple`, `feature-distribution`, and `mixed`.

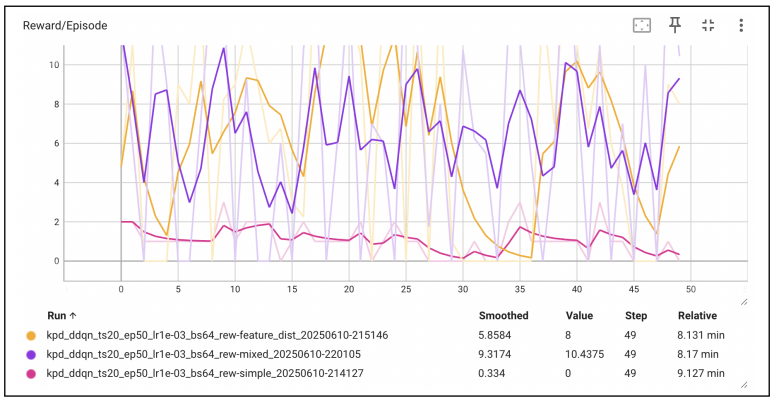

Figure 5: Reward per episode for DDQN with 20 timesteps over 50 episodes under reward functions `simple`, `feature-distribution`, and `mixed`.

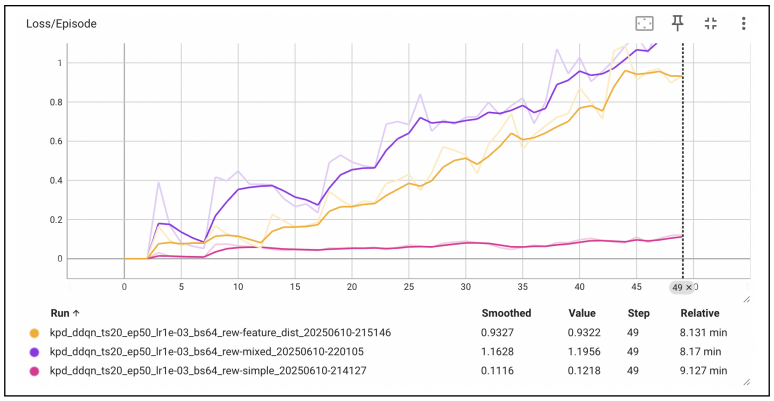

Figure 6: Loss per episode for DDQN with 20 timesteps over 50 episodes under reward functions `simple`, `feature-distribution`, and `mixed`.

6 Discussion

In section 5.2, we discussed the DDQN algorithm's failure to learn, and considered several factors that may be contributing to this and ways to address them. Future research should explore these options. One challenge, however, is that several include increasing the amount of data accessed by the agent and used in each step of the algorithm; this demands an increased level of computing power. Yet our state space and action space are quite large; as we also noted in section 5, running the algorithm on trajectories of 300 timesteps or more requires relatively high computing time. This suggests a need to try to further optimize the efficiency of the DDQN algorithm or, if need be, trim the state space and action space as appropriate (for example, when not running a fairness-informed reward function, we can ignore the demographic features of each patient-donor pair).

Another challenge, as previously mentioned, is the complexity of the real KPD environment in comparison to the our simulated dataset and trained model, and the many ways in which our data and model fall short in capturing causalities, dependencies, and relevant information. We don't know yet how meaningful each shortcoming will be for the validity of our outcomes, and it will be challenging to detect these shortcomings, as they will likely be obscured by the ways in which our simulated datasets convenience the training algorithm and yield success where it may not be due.

We also find it important to highlight that the definition of fairness used in this research attempts to sway the distributions of matched patients and donors to meet a pre-determined desired distribution. This prescriptive approach to fairness carries great responsibility for the policymaker implementing this algorithm, and is liable for abuse if misused. Future research may consider the various consequences of the mismatch between the KPD simulation and model and the real KPD matching market, as well as the range of implications of policymakers' choices of desired feature distributions.

7 Conclusion

Our use of a DDQN algorithm with a GNN model to address the dynamic KPD matching problem presents a promising avenue for leveraging deep reinforcement learning to solve complex online matching problems with multiple (possibly competing) objectives. Our results showed that DDQN performs competitively with the recent ADP algorithm developed by You and Vossen (2024). In our experiments, we did not manage to catalyze learning, so the natural future direction involves continuing to tune hyperparameters, as well as re-evaluate the design of our reward functions and the architecture of our model. One particularly challenging issue for future research to address is the computing overhead of our large state space, action space, and model parameters. Once the model is able to learn, we can explore fairness for various features, detect subtle tradeoffs, and even experiment with multi-group fairness over several features. The opportunities to train this model to achieve different complex objectives are innumerable.

8 Team Contributions

- **Odelia Lorch:** I completed this work myself.

References

Mohammad Akbarpour, Shengwin Li, and Shayan Oveis Gharan. 2020. Thickness and information in dynamic matching markets. *Journal of Political Economy* 128, 3 (2020), 783–815. https://doi.org/10.1086/704761

Rémi Castera, Felipe Garrido-Lucero, Mathieu Molina, Simon Mauras, Patrick Loiseau, and Vianney Perchet. 2024. The Price of Fairness in Bipartite Matching. *CoRR* abs/2403.00397 (2024). arXiv:2403.00397 [cs.GT] https://arxiv.org/abs/2403.00397

Rajkiran Deshpande. 2024. Smart match: revolutionizing organ allocation through artificial intelligence. *Frontiers in Artificial Intelligence* Volume 7 - 2024 (2024). https://doi.org/10.3389/frai.2024.1364149

John P Dickerson, Ariel D Procaccia, and Tuomas Sandholm. 2013. Failure-aware kidney exchange. In *Proceedings of the fourteenth ACM conference on Electronic commerce*. 323–340.

John Paul Dickerson and Tuomas Sandholm. 2014. Balancing efficiency and fairness in dynamic kidney exchange.

Seyed Esmaeili, Sharmila Duppala, Davidson Cheng, Vedant Nanda, Aravind Srinivasan, and John P. Dickerson. 2023. Rawlsian Fairness in Online Bipartite Matching: Two-Sided, Group, and Individual. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 5 (2023). https://doi.org/10.1609/aaai.v37i5.25698

S. M. Flechner, A. G. Thomas, M. Ronin, J. L. Veale, D. B. Leeser, S. Kapur, J. D. Peipert, D. L. Segev, M. L. Henderson, A. A. Shaffer, M. Cooper, G. Hil, and A. D. Waterman. 2018. The first 9 years of kidney paired donation through the National Kidney Registry: Characteristics of donors and recipients compared with National Live Donor Transplant Registries. *American Journal of Transplantation* 18, 11 (Nov. 2018), 2730–2738. https://doi.org/10.1111/ajt.14744 Epub 2018 Apr 30.

Irena Gao. 2019. Fair Matching in Dynamic Kidney Exchange. arXiv:1912.10563 [cs.GT]

Neetika Garg, Carrie Thiessen, Peter P. Reese, Matthew Cooper, Ruthanne Leishman, John Friedewald, Asif A. Sharfuddin, Angie G. Nishio Lucar, Darshana M. Dadhania, Vineeta Kumar, Amy D. Waterman, and Didier A. Mandelbrot. 2024. Temporal trends in kidney paired donation in the United States: 2006-2021 UNOS/OPTN database analysis. *American Journal of Transplantation* 24, 1 (01 Jan 2024), 46–56. https://doi.org/10.1016/j.ajt.2023.09.006

C. M. Holscher, K. Jackson, E. K. H. Chow, A. G. Thomas, C. E. Haugen, S. R. DiBrito, C. Purcell, M. Ronin, A. D. Waterman, J. Garonzik Wang, A. B. Massie, S. E. Gentry, and D. L. Segev. 2018. Kidney exchange match rates in a large multicenter clearinghouse. *American Journal of Transplantation* 18, 6 (June 2018), 1510–1517. https://doi.org/10.1111/ajt.14689 Epub 2018 Mar 9.

Hadi Hosseini, Zhiyi Huang, Ayumi Igarashi, and Nisarg Shah. 2023. Class Fairness in Online Matching. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 5 (2023). https://doi.org/10.1609/aaai.v37i5.25704

David B. Leeser, Alvin G. Thomas, Ashton A. Shaffer, Jeffrey L. Veale, Allan B. Massie, Matthew Cooper, Sandip Kapur, Nicole Turgeon, Dorry L. Segev, Amy D. Waterman, and Stuart M. Flechner. 2020. Patient and Kidney Allograft Survival with National Kidney Paired Donation. *Clinical Journal of the American Society of Nephrology* 15, 2 (Feb. 2020), 228–237. https://doi.org/10.2215/CJN.06660619

Will Ma, Pan Xu, and Yifan Xu. 2023. Fairness maximization among offline agents in online-matching markets. *ACM Transactions on Economics and Computation* 10, 4 (2023), 1–27. https://doi.org/10.1145/3569705

National Kidney Registry. 2022/2024. National Kidney Registry Website. https://www.kidneyregistry.com/. Accessed: 2025-05-27.

David B. Olawade, Sheila Marinze, Nabeel Qureshi, Kusal Weerasinghe, and Jennifer Teke. 2025. The impact of artificial intelligence and machine learning in organ retrieval and transplantation: A comprehensive review. *Current Research in Translational Medicine* 73, 2 (2025), 103493. https://doi.org/10.1016/j.retram.2025.103493

Govind S. Sankar, Anand Louis, Meghana Nasre, and Prajakta Nimbhorkar. 2021. Matchings with Group Fairness Constraints: Online and Offline Algorithms. arXiv:2105.09522 [cs.DS]

D. L. Segev, S. E. Gentry, R. A. Montgomery, S. Kulkarni, G. Chen, and D. G. Warnock. 2023. Temporal Trends in Kidney Paired Donation in the United States: 2006–2021 UNOS/OPTN Database Analysis. *American Journal of Transplantation* (2023). Sep 2023.

William St-Arnaud. 2021. Towards fairness in Kidney Exchange Programs.

Stanislaw M Stepkowski, Beata Mierzejewska, David Fumo, Dulat Bekbolsynov, Sadik Khuder, Caitlin E Baum, Robert J Brunner, Jonathan E Kopke, Susan E Rees, Connie Smith, et al. 2019. The 6-year clinical outcomes for patients registered in a multiregional United States Kidney Paired Donation program-a retrospective study. *Transplant International* 32, 8 (2019), 839–853.

Fan You and Thomas Vossen. 2024. An Approximate Dynamic Programming Approach to Dynamic Stochastic Matching. *INFORMS Journal on Computing* 36 (02 2024). https://doi.org/10.1287/ijoc.2021.0203

A Construction of Dataset

The statistical distributions and rates used to simulate our dataset are primarily from the 2006-2021 UNOS/OPTN database as analyzed in Segev et al. (2023), as well as the National Kidney Registry's (NKR) Quarterly Report from August 2024 National Kidney Registry (2024) and the U.S. Transplant Candidate Self-Reported Health Survey .

We simulated the dataset at each timestep as a randomized graph with a fixed arrival rate $\lambda = 10$ pairs per day (National Kidney Registry (2024)), and departure rate calculated from the median waiting in the market before leaving (without matching) is ≈ 200 days (National Kidney Registry (2024)), and the rest of the nodes' data determined according to the distributions in Table 4.

In table 4, two notable features are `incompat-reason` and `antigens`. `incompat-reason` is dependent on `patient-blood`, `donor-blood`, `patient-cpra`, and `antigens`. If `patient-blood` and `donor-blood` are ABO incompatible, the `incompat-reason` is "ABO." If instead they are ABO compatible, but patient antibodies (`patient-cpra`) and donor antigens (`antigens`) are HLA incompatible, then the `incompat-reason` is "HLA." If instead they are HLA compatible, then they are considered compatible, and `incompat-reason` is "compatible." This mirrors the logic of UNOS/OPTN and NKR. For `antigens`, we found no statistical summary for KPD donor candidates, so we defined a generic list of common HLA antigens and treated the distribution uniformly.

B Approximate Dynamic Programming Algorithm

The Approximate Dynamic Programming (ADP) algorithm devised by You and Vossen (2024) represents that patient-donor pairs matching pool as an evolving KPD graph state G_t, where each G_t encodes the features of each patient–donor pair (for example compatibility, patient medical state, demographic information), as well as the pool size and composition at decision epoch t.

The algorithm begins by choosing basis functions $\{\phi_j(G_t)\}_{j=1}^{J}$ over G_t, and then approximates the the cost-to-go at time t as:

$$\hat{V}(G_t;\theta) = \sum_{j=1}^{J} \theta_j \, \phi_j(G_t).$$

where θ_j are weights to be decided by the linear program. They then formulate an approximate linear program using the Bellman Inequality to enforce that the estimated value respects dynamic matching rewards:

$$\begin{aligned} \min_{\theta} \quad & \sum_{G_t} \mu(G_t)\, \hat{V}(G_t;\theta) \\ \text{s.t.} \quad & \hat{V}(G_t;\theta) \;\geq\; R(G_t, M) + \gamma \mathbb{E}_{G_{t+1}}[\hat{V}(G_{t+1};\theta)] \\ & \quad \forall\, G_t,\ \forall\, M \in \mathcal{M}(G_t). \end{aligned}$$

Table 4: Statistical Distributions of Patient-Donor Features

Feature	Distribution		Source
`patient-gender`	Male	61.6 %	2006-2021 UNOS/OPTN Database
	Female	38.4 %	
`patient-race`	White	72.7 %	2006-2021 UNOS/OPTN Database
	Black/African American	12.7 %	
	Hispanic/Latino	9.5 %	
	Asian	3.8 %	
	Other/Unknown	1.3 %	
`patient-age`	< 30:	8.5 %	2006-2021 UNOS/OPTN Database
	30–44:	29.1 %	
	45–59:	42.5 %	
	60–74:	17.1 %	
	≥ 75:	2.8 %	
`patient-blood`	Type O	41.4 %	2006-2021 UNOS/OPTN Database
	Type A	28.1 %	
	Type B	17.5 %	
	Type AB	13.0 %	
`donor-gender`	Male	35.7 %	2006-2021 UNOS/OPTN Database
	Female	64.3 %	
`donor-race`	White	66.8 %	2006-2021 UNOS/OPTN Database
	Black/African American	14.7 %	
	Hispanic/Latino	10.9 %	
	Asian	4.2 %	
	Other/Unknown	3.4 %	
`patient-age`	< 30:	11.9 %	2006-2021 UNOS/OPTN Database
	30–44:	44.3 %	
	45–59:	33.5 %	
	≥ 60:	13.6 %	
`patient-blood`	Type O	39.8 %	2006-2021 UNOS/OPTN Database
	Type A	31.9 %	
	Type B	19.2 %	
	Type AB	9.1 %	
`incompat-reason`	ABO	blood types incompatible	2006-2021 UNOS/OPTN Database
	HLA	cPRA-HLA incompatible	
	compatible	otherwise	
`patient-health`	Good	50%	U.S. Transplant Candidate
	Fair	35%	Self-Reported Health Survey
	Poor	15%	
`patient-cpra`	cPRA 0–20 %:	54.9 %	2006-2021 UNOS/OPTN Database
	cPRA 21–80 %:	23.5 %	
	cPRA 81–98 %:	14.2 %	
	cPRA 99–100 %:	7.5 %	
`antigens`	common HLA antigens (A1, A2, A3, A11, B7, B8, B27, B44, DR1, DR4, DR7, DR15, DQ2, DQ6, DQ8)	uniform	N/A

where $R(G_t, M)$ is number of transplants (or weighted matches) executed by matching action M in graph G_t; $\mathcal{M}(G_t)$ is the set of feasible match cycles/chains given G_t; and the transition $G_t \to G_{t+1}$ models arrivals, exits (matched or ineligible), and updates of patient states.

The algorithm samples graph states and matching actions from the simulated dataset (described in Section 3.1 and Appendix A). It then selects high-variance or high-impact states (e.g., near capacity thresholds or with many hard-to-match pairs) to yield a tractable LP capturing key dynamics of dynamic KPD matching.

The algorithm then solves the reduced LP to obtain θ^*. Then, at each decision epoch t, the policy returns:

$$M_t^* = \arg \max_{M \in \mathcal{M}(G_t)} \Big\{ R(G_t, M) + \gamma \mathbb{E}_{G_{t+1}}[\hat{V}(G_{t+1}; \theta^*)] \Big\},$$

executing the cycle M_t^*.

Finally, the algorithm iteratively refines the policy over multiple samples by sampling the distribution based on observed visitation frequencies and mismatches and updating the basis functions. By doing so, it can incorporate emergent compatibility patterns, or resample states emphasizing poorly approximated regions, thus producing a more robust matching policy.