DECAF: Learning to be Fair in Multi-agent Resource Allocation

Ashwin Kumar
ashwinkumar@wustl.edu
Washington University in St. Louis

William Yeoh
wyeoh@wustl.edu
Washington University in St. Louis

Abstract

A wide variety of resource allocation problems operate under resource constraints that are managed by a central arbitrator, with agents who evaluate and communicate preferences over these resources. We formulate this broad class of problems as *Distributed Evaluation, Centralized Allocation (DECA)* problems and propose methods to learn fair and efficient policies in centralized resource allocation. Our methods are applied to learning long-term fairness in a novel and general framework for fairness in multi-agent systems. We show three different methods: (1) A joint weighted optimization of fairness and utility, (2) a split optimization, learning two separate Q-estimators for utility and fairness, and (3) an online policy perturbation to guide existing black-box utility functions toward fair solutions. Through experiments on multiple domains, we compare these methods and discuss relevant use cases for each of them. We also highlight an important and overlooked factor in learning to act fairly in constrained multi-agent problems: The importance of past-discounting and warm starts for learning fair behavior.

1 Introduction

AI has improved the capabilities of many modern systems, becoming an integral part of most automated solutions to hard problems. Increasingly, AI algorithms are being used for decision making, with the potential to impact millions of people. In multi-agent settings, the typical approach is to take a utilitarian objective, optimizing the cumulative utility over all individuals. However, this may lead to undesirable biases in decision making, necessitating fair AI algorithms.

We look at a class of problems that we term *Distributed Evaluation, Centralized Allocation (DECA)*. To the best of our knowledge, thus far, researchers have investigated the different problems in this class separately and applied domain-specific approaches to solve them (Shah et al., 2020; Qin et al., 2022; Kube et al., 2019; 2023). In this paper, we take the first step towards representing them with a single DECA formulation and proposing fairness approaches that generally apply broadly to all DECA problems.

In DECA, multiple agents act in an environment, while a central controller coordinates their behavior to satisfy resource and environment constraints. The overall system objective is an aggregation of agent utilities, while each agent's goal is to maximize individual utility. In this system, each agent performs local evaluations of their available actions (DE), and the central controller aggregates these evaluations, optimizes for maximization of system utility and ensures constraint satisfaction before allocating actions to agents (CA). These problems typically have a temporal nature, with agents exiting and re-entering the system at different times, as well as having variable action spaces. This makes DECA a computationally challenging and interesting class of problems as well.

In this work, we tackle the important problem of improving long-term fairness in DECA problems. Algorithmic bias is a growing concern where automated algorithms are used for decision-making (Mehrabi et al., 2021). Ensuring fairness helps mitigate algorithmic bias and promotes user trust in automated systems. Further, fair resource allocation may also be desirable from the controller's perspective. The general solution for DECA problems involves having a utility estimate for each agent's actions, followed by solving a constrained optimization to compute the best actions for each agent. We propose adding a fairness estimate for each agent's actions as well, to guide the central controller towards a fairer allocation through combining the fairness and utility estimates. We propose three ways to do this:

1. **Joint**: Learning a joint estimator for fairness and utility as a multiobjective optimization.
2. **Split**: Learning separate estimators, where the tradeoff weight between fairness and utility can be controlled online during execution.
3. **Fair-Only**: Learning a separate fairness estimate that modifies a given black-box utility function.

We compare the efficacy of these methods, showing each of their strengths in different scenarios. We also demonstrate how the Split and Fair-Only approaches allow online tuning of the fairess-utility tradeoff, which is important for real-world applications. This has been overlooked by previous approaches for learning fair-efficient policies in multi-agent RL. We also highlight a key distinguishing factor in our approach: The use of warm starts and past discounting for learning fairness.

2 Related work

While DECA has not been formalized in prior work, it has seen application in many domains. From optimizing passenger-driver matches in ridesharing (Shah et al., 2020; Qin et al., 2022) to efficient allocation of homelessness resources (Kube et al., 2019; 2023), many real-world applications follow this general structure. It is also analogous to the predict-then-optimize (P+O) approach (Wang et al., 2021; Elmachtoub & Grigas, 2022), where a predictive model estimates unknown parameters that are subsequently used in optimization. However, unlike P+O, which may not specifically address resource allocation or multi-agent systems, DECA explicitly focuses on these complexities. This distinction is crucial as it allows us to restrict the problem space and tailor our research towards enhancing fairness within multi-agent resource allocation.

Significant research has addressed algorithmic bias, where ML models, such as those used in hiring decisions (Raghavan et al., 2020), can exhibit harmful biases. We refer readers to an extensive survey by Mehrabi et al. (2021) for a review of recent work. These studies typically focus on debiasing the outputs of predictive models to meet fairness criteria such as equalized odds (Hardt et al., 2016) or demographic parity (Dwork et al., 2012). However, our work diverges from this approach. Instead of correcting biases in predictions, our objective is to develop algorithms that inherently promote fair decision-making through the actions they optimize.

For this paper, our focus is on designing ways to learn fair policies in a multi-agent reinforcement learning setting. Gajane et al. (2022) present a survey on RL methods used to improve fairness. We now highlight a few papers which are the closest to our work: FEN (Jiang & Lu, 2019) is an approach for learning a hierarchical fair-efficient policy for multi-agent coordination. However, the model does not allow for resource constraints, instead opting for a first-come-first-serve approach. Further, this approach needs communication between agents to allow agents to choose between acting fairly and efficiently. Zimmer et al. (2021) and Siddique et al. (2020), on the other hand, propose to optimize fairness in a multi-objective MDP, where each agent's utility is treated as a separate objective, and the goal is to optimize the a social welfare function over agent rewards. This means the learning agent has to predict the utility over the joint action space (Siddique et al., 2020), or use a decentralized policy gradient based approach (Zimmer et al., 2021), which prevents use of global constraints. Simple Incentives (Kumar et al., 2023) is an approach for improving fairness in rideshare-matching that attempts to improve fairness through myopic fairness post-processing of black-box utility estimates. However, it does not attempt to learn long-term fairness.

The DECA approach allows us to consider global constraints while allocating resources, opening up the scope for better global solutions, which none of the prior approaches allow. The distributed evaluation allows each agent to only learn a local value function, which reduces the complexity when compared to learning a joint policy. Further, our Split and Fair-Only approaches allow changing the tradeoffs between utility and fairness post-training, which provides additional flexibility that previous approaches lack. Kumar et al. (2023) provides this flexibility. However, it only works with black-box utility functions and only provides myopic improvements. Thus, it is subsumed by our Only-fair approach, which learns the long-term fairness for black-box utility functions.

3 Problem formulation

In the context of Distributed Evaluation, Centralized Allocation (DECA), our primary goal is to integrate fairness into the decision-making process of resource allocation in multi-agent systems. Formally, we seek to maximize a combined measure of system utility and fairness, represented as:

$$\max \mathcal{U}_T + \beta \mathcal{F}_T \tag{1}$$

where $\mathcal{U}_T$ denotes the total utility at time T and $\mathcal{F}_T$ represents the fairness measure, weighted by β.

In this section, we describe the DECA optimization framework and how it can be used for resource allocation. In the next section, we will describe DECAF, our solution to learn to improve fairness in this framework.

3.1 Distributed Evaluation, Centralized Allocation (DECA)

We define the DECA framework through a temporal resource allocation problem formulated as a constrained Multiagent Markov Decision Process (De Nijs et al., 2021). Our model is described by the tuple $\langle \alpha, S, \{A_i\}_{i \in \alpha}, T, \{R_i\}_{i \in \alpha}, \gamma, c \rangle$, where:

- α is the set of agents indexed by i.
- S is the global state space.
- A_i is the action space for agent i.
- $T : S \times A_1 \times A_2 \times \ldots \times A_n \times S \rightarrow [0, 1]$ represents the joint transition probabilities.
- $R_i : S \times A_i \rightarrow \mathbb{R}$ denotes the reward function for agent i.
- γ is the discount factor for future rewards.
- $c : A_1 \cup A_2 \cup \ldots \cup A_n \rightarrow \mathbb{R}^K$ maps each action to its resource consumption.

In a DECA problem, agents independently evaluate actions based on their local states (DE), while a central controller aggregates these evaluations and optimizes resource allocation subject to constraints (CA). These problems often involve dynamic agent populations and fluctuating action spaces, which adds computational complexity and richness to the model.

3.2 Optimization framework

The distributed evaluation (DE) step involves agents learning to predict the utilities of state-action pairs using approaches such as Deep Q-learning (Mnih et al., 2013; Hasselt et al., 2016). The utility estimates are computed using partially-observable post-decision states, which are estimated locally, ignoring other agents' actions due to the infeasibility of exploring the joint state space.

The central allocation (CA) step solves a mixed-integer program that combines predicted utilities and resource constraints. Let $\mathcal{A}$ denote the allocation of actions decided by the central allocator such that $\mathcal{A}_i$ is the action assigned to agent i, and let there be K types of resources, each represented by $k \in \{1, 2, \ldots, K\}$, such that the number of resources of resources available can be written as $\mathcal{R} \in \mathbb{R}^K$. We thus say that each resource k has an availability $\mathcal{R}_k$. This gives us the following optimization:[1]

$$\max_{x_i(a)\in\{0,1\}} \sum_{i\in\alpha}\sum_{a\in A_i} x_i(a)Q(s_i^a) \tag{2}$$

subject to:

$$\sum_{a\in A_i, x_i(a)\in\{0,1\}} x_i(a) = 1, \quad \forall i \in \alpha, \quad \text{(Action Constraint)} \tag{3}$$

$$\sum_{a\in\mathcal{A}} c(a)_k \leq \mathcal{R}_k, \quad \forall j \in \{1,\ldots,K\}, \quad \text{(Resource Constraint)} \tag{4}$$

These constraints ensure that each agent is assigned exactly one action and that total resource usage does not exceed available supplies. The ILP described above forms the central controller, and the Q-value estimator controls distributed evaluation. This benefits over completely distributed approaches by encapsulating resource constraints, and over completely centralized approaches by reducing the complexity of the learning objective. This setup is seen in many resource allocation problems (Kube et al., 2019; Alonso-Mora et al., 2017; Shah et al., 2020).

4 DECAF: Fairness in DECAs

Our approach to fairness in DECA systems involves balancing fairness and utility through adjustments in the policy formulation. To achieve this, we target the DE step, modifying Q to be an estimator of the combined fair-efficient objective, with a weight $\beta \geq 0$ used to regulate relative value of fairness and utility. Let $R_u(s_i, a)$ and $R_f(s_i, a)$ denote the immediate utility and fairness return of agent i taking action a, and let θ denote the parameters of the Q-function. We want to minimize the loss function $J_\theta(\pi) = \mathbb{E}_{\tau\sim\pi,\theta}L(\delta(\tau))$, where $\delta_\theta(\tau)$ is the Bellman error of the transition $\tau = \langle s, a, s'\rangle$. For this discussion, let $L(\cdot)$ denote mean-squared loss function. We propose three approaches for integrating fairness:

Joint Optimization (JO): A single estimator optimizes a weighted combination of fairness and utility.

$$\delta(\tau) = R_u(s,a) + \beta R_f(s,a) + \gamma Q(s') - Q(s^a) \tag{5}$$

Split Optimization (SO): Separate estimators for fairness ($F(\cdot)$) and utility ($U(\cdot)$) allow dynamic adjustment of their trade-off during policy execution.

$$\delta^f(\tau) = R_f(s,a) + \gamma F(s') - F(s^a) \tag{6}$$

$$\delta^u(\tau) = R_u(s,a) + \gamma U(s') - U(s^a) \tag{7}$$

$$Q(s^a) = U(s^a) + \beta F(s^a) \tag{8}$$

Fair-Only Optimization (FO): A fairness estimator ($F(\cdot)$) adjusts a pre-existing utility function $U^*(\cdot)$ to incorporate fairness, useful when utility functions are provided externally.

$$\delta^f(\tau) = R_f(s,a) + \gamma F(s') - F(s^a) \tag{9}$$

$$Q(s^a) = U^*(s^a) + \beta F(s^a) \tag{10}$$

Each method modifies the Q-function, which in turn influences the central controller's decisions. SO and FO offer the additional benefits of interpretability, as during execution, we are able to discern how much of the decision was based on the utility gain and fairness improvement.

SO also provides some useful properties described below. The full proofs are included in the appendix.

[1] Throughout the paper, s^a represents the local post-decision state computed by the agent, while s' is the true next state.

Theorem 1. *Given perfect estimates for utility and fairness, increasing β always improves the one-step fairness gain for SO with $\gamma = 0$.*

Theorem 2. *For a large enough β, the fairest allocation will be selected with perfect utility and fairness estimators for SO with $\gamma = 0$.*

We also provide the corollaries to these theorems for improving utility as β is reduced (see the appendix).

4.1 Immediate fairness return

So far, we have discussed how we will optimize for fairness, but we have not yet specified what fairness means in our context. In this work, we consider fairness as the variance over a set of agent metrics $\mathbf{Z} = \{z_i\}_{i\in\alpha}$. The agent metric z_i is typically the accumulated agent utility, but can be other values as well. Let $\mathbf{Z}_t^\pi$ represent the distribution of agent metrics at time t following policy π. Our fairness objective is to minimize the variance at the terminal step, $\operatorname{argmax}_\pi \mathcal{F}_T = -\operatorname{var}(\mathbf{Z}^\pi_{t=T})$.

We derive $R_f(s, a)$, the signal for variance minimization, defining it as the immediate fairness reward obtained by an agent's action. Consider a joint action A^t, and the distributions Z_t and Z_{t+1}. Let $\bar{z}_t$ denote the average agent metric at time t. Then, we calculate the change in the fairness metric because of this action as:

$$\begin{aligned}
\Delta\mathcal{F}|A^t &= \mathcal{F}_{t+1} - \mathcal{F}_t && (11)\\
&= -\operatorname{var}(\mathbf{Z}_{t+1}) + \operatorname{var}(\mathbf{Z}_t) && (12)\\
&= -\frac{1}{n}\sum_{i\in\alpha}\left(z_i^{t+1} - \bar{z}_{t+1}\right)^2 + \frac{1}{n}\sum_{i\in\alpha}\left(z_i^t - \bar{z}_t\right)^2 && (13)
\end{aligned}$$

To assign the contribution of any one agent, we can simply extract the relevant terms:

$$R_f(i, A_i^t) = -\frac{1}{n}\left(\left(z_i^{t+1} - \bar{z}_{t+1}\right)^2 - \left(z_i^t - \bar{z}_t\right)^2\right) \tag{14}$$

5 Experiments

5.1 Environments

We adapt the environments used by Jiang & Lu (2019) to make them compatible with the DECA framework. We also introduce a new environment, called BiasedDM, in which the decision-maker is biased and has differing utilities for different agents. We describe each environment below.

Matthew: This environment is intended to exhibit the Matthew effect (Rigney, 2010; Gao et al., 2023), where the rich get richer and the poor get poorer. We have ten agents moving in a continuous unit grid, with three resources available at any time. Eating resources gives agents a speed and size boost, letting them reach newer resources faster. Some agents start bigger and faster, giving them an advantage. An action consists of assigning an agent to a resource, following which other agents cannot pick it up. Alternatively, agents can take the null action, and move randomly in some direction. The agents send their utility estimates for each available action to the decision-maker, which allocates available resources subject to the constraint that no two agents can be assigned to the same resource. We abstract away the motion planning by making agents always move in a straight line towards their target.

Job: In this environment, four agents operate on a discretized grid, with one fixed square containing a job. Agents get a reward if they are occupying the job's location. The resources in this case are the grid locations, with the constraint that only one agent may occupy a location at any time. Agents' actions are to move in cardinal directions, and they communicate their preferences over directions to the decision maker, which assigns the final moves.

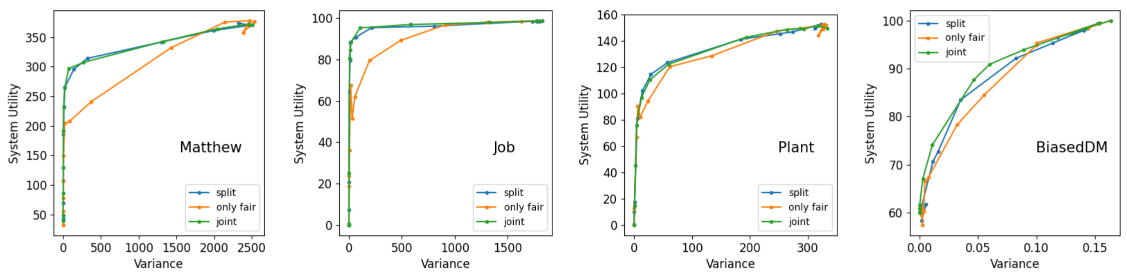


Figure 1: Change in system utility and fairness as β is increased. The line starts at $\beta = 0$ (top right), and each successive point is a logarithmic increase in β, ending at the highest β in the bottom-left. For all domains, we can see that split and joint optimization perform similarly, while learning only fairness leads to different behavior. Each point depicts the average performance over five different models trained at that β value.

Plant: In this environment, five agents live on a discretized grid, where three kinds of resources may spawn. There are eight resources on the grid at any time. Each agent has a unique combination of resources they must obtain to construct one 'unit' and earn a reward. Additionally, the requirements are harder for some agents to fulfill than others. The agents compute preferences over available resources, and the decision maker assigns agents to resources such that at most one agent is assigned to each resource. The agents deterministically move towards their assigned resource.

BiasedDM: Unlike the other environments where fairness is assessed by variance in agent rewards and overall utility by the sum of agent utilities, this environment introduces an explicit bias in decision-making. Here, five agents compete for a single resource at each timestep, with the resource's utility to the decision-maker disproportionately larger for higher-indexed agents ($0.2 \times i$ for agent i). Optimal utility is achieved by consistently allocating the resource to agent 5. Fairness is evaluated based on the resource distribution over time, reflecting a disconnect between fairness and utility.

5.2 Experiment setup

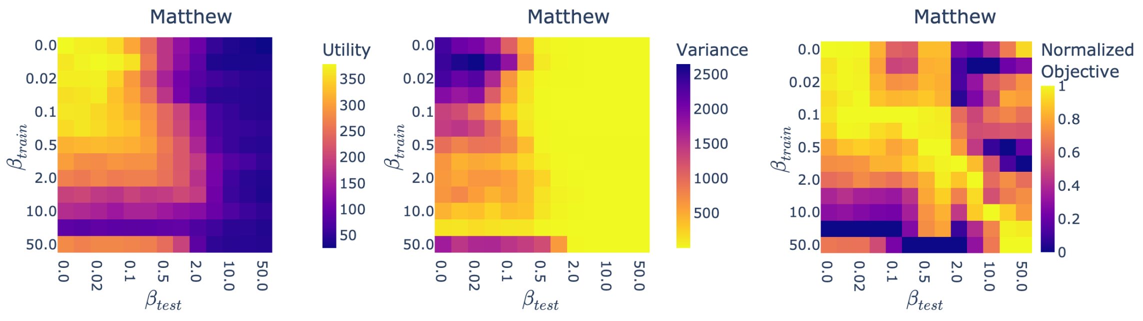
We conduct experiments for maximizing the objective in Eq. 1, where the system utility is the sum of all agent utilities at the end of an episode, and the fairness is measured as the negative of the variance of agent resources at the end of the episode. We perform experiments for a variety of β values, repeating each configuration 5 times for each of our three settings: **Joint Optimization (JO), Split Optimization (SO)** and **Fair-Only Optimization (FO)**. The domains were hand-crafted to fit with the DECA framework. Each model uses the same network architecture, with two hidden layers of dimension 20, and output of dimension 1. The utility model used for FO is randomly selected from the JO models trained with $\beta = 0$. We included features indicating the relative advantage of each agent as a signal for fairness, in addition to the features describing the local observation of each agent.

6 Results

Figure 1 shows the performance of all three DECAF methods (JO, SO, FO) on the four domains discussed above. For each method, we varied the hyperparameter β controlling the fairness-utility tradeoff (Eqs. 5, 8, 10) in a logarithmic manner. Each line in the figures traces the performance starting with $\beta = 0$ (top-right), increasing to the highest β value (bottom-left).

6.1 Efficacy of the fairness-utility optimization

For all domains, all three methods are able to learn expressive policies which lie at various points close to the Pareto front. This shows that the optimization problem allows the model to trade off



(a) System Utility (b) Variance (c) Normalized Objective

Figure 2: Evaluation of SO models trained on β_{train} and evaluated on β_{test} for the Matthew environment. Brighter colors indicate better outcomes. The "Normalized Objective" plot scales the values in each column of the heatmap between 0 and 1.

utility and fairness to show diverse behaviors as required by the user. This also confirms that the fairness reward proposed for minimizing variance is a good signal.

6.2 Comparison of methods

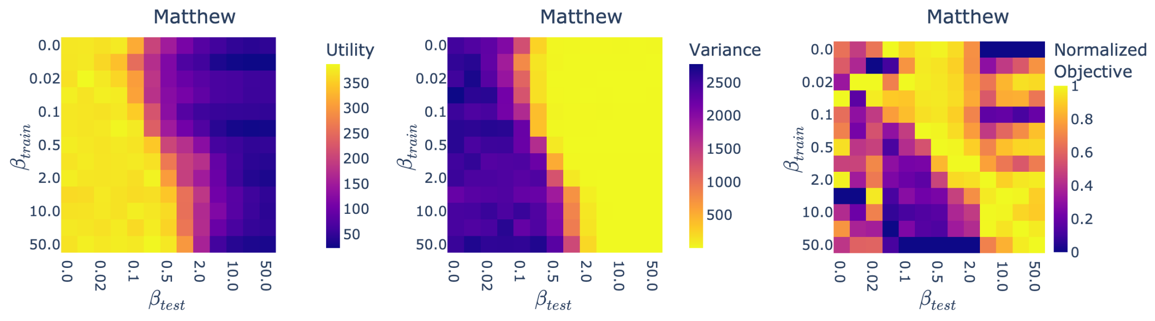
In our results, JO and SO generally exhibit similar performance characteristics, suggesting that simultaneous evolution of utility and fairness estimates is beneficial. In contrast, the performance of FO varies significantly across different environments and depends heavily on the quality of the initial black-box utility model. FO is consistently Pareto-dominated by both JO and SO. This underperformance is likely due to out-of-distribution transitions for the fixed utility model, which are more problematic in FO when a large fairness weight (β) is used, causing a larger shift in the state distribution and resulting in degraded utility estimates.

6.3 Generalization using Split Optimization (SO)

Figure 2 shows detailed results for the Matthew environment, when using SO. We evaluate each model trained on a particular β_{train} on all other β_{test}. This allows us to see how well the trained fairness and utility models generalize when the operating β is changed. Note that for all these models, β is not provided as an input to the Q function.

The diagonal elements show the behavior when training and testing is done on the same β value. From the plots for system utility (Figure 2a) and variance (Figure 2b), we can see that as β_{test} increases, variance improves, and as β_{test} decreases, utility improves. This is the expected behavior, and the major advantage of SO over JO. With JO, the model only predicts a single value, so we are unable to change the tradeoff weight during evaluation. Thus, we need a unique model for each β that we want the model to work for. However, with SO, selecting just a few spread out β values can allow us to extrapolate between them, providing online adaptability.

The third heatmap (Figure 2c) shows the normalized objective value (Eq. 1 evaluated using β_{test}). For each column, we normalize the objective value by scaling it between 0 and 1. Thus, the brightest colors show the models that provide the best objective value for a particular β_{test}. The figure shows that these models fall close to the diagonal. Thus, for β_{test} that are close to β_{train}, the models perform near-optimally, while for larger differences, performance may be worse. Note that each β is logarithmically larger than the previous. This shows that SO has the flexibility to function well at operating points away from the β_{train} that it is trained for. These observations hold for other domains as well, and the results are included in the Appendix.



(a) System Utility (b) Variance (c) Normalized Objective

Figure 3: Evaluation of FO models trained on β_{train} and evaluated on β_{test} for the Matthew environment. Brighter colors indicate better outcomes. The "Normalized Objective" plot scales the values in each column of the heatmap between 0 and 1.

6.4 Effectiveness of Fair-Only Optimization (FO)

Like SO, FO is also able to generalize well when different β_{test} are used to evaluate the learned models (Figure 3). Because the utility model is fixed, all models achieve high utility as $\beta_{test} \to 0$ (Figure 3a). Further, all models also improve fairness as β_{test} grows larger (Figure 3b). The behavior change from utility-oriented to fairness-oriented is much sharper in FO when compared to SO. Looking at the normalized objective (Figure 3c), we see that close to the diagonal, we have the best performance, and as we deviate further from β_{train}, the performance degrades. Despite being Pareto-dominated by SO and JO at intermediate β values, FO has the advantage of reliability: A trusted black-box utility model can be used in conjunction with a possibly smaller fairness model, with the guarantee to behave optimally as β_{test} is reduced.

6.5 On the importance of past discounting and warm starts

We also highlight two departures from prior work in our implementation of DECAF for learning fairness: (1) We discount the agent metrics $\mathbf{Z}$ over the past, and (2) we implement warm starts for initializing $\mathbf{Z}$ instead of initializing at 0. The past discounts are necessary to account for the time dependence of various normalized fairness metrics, like the Gini coefficient and coefficient of variation, or the variance over normalized agent metrics. In these cases, actions taken early on can cause larger changes to the fairness metric while actions taken after a sufficient history has been established have an imperceptible effect. This is undesirable in long-horizon settings, but can be remedied by past discounts, effectively 'forgetting' events that happened far in the past. The warm starts function to counteract the other pitfall in most fairness metrics: The zero vector is perfectly fair, and any changes can incur a large fairness penalty. Adding randomized warm starts helps in preventing the algorithm from converging to this trivial solution. In practice, we keep the warm start values small, so that the past discounts effectively scale them down to zero over the course of an episode.

7 Conclusions and future work

We proposed DECAF, a framework to learn to learn long-term utility and fairness estimates for multi-agent resource allocation. Our approach is one of the first that allows for optimized fair resource allocation under resource constraints. Our approach also does not necessitate that the fairness and utility metrics be directly aligned, which allows for increased diversity in problem settings. Split and Fair-Only optimization also allow online trade-offs between utility and fairness weights without retraining, while also improving the interpretability of the allocation. Our results show how each of our approaches allows for improved flexibility in different scenarios.

We also discuss some limitations of our approach. Our research currently only works with Q-learning. Deriving policy learning for DECA problems is hard, and a good avenue for future research. The primary challenge associated with policy learning is the dynamic state and action space that can be encountered in many resource allocation problems, as well as the indirect relation between local agent 'policies' and the final action resulting from the ILP optimization. Our theoretical results hold for $\gamma = 0$, but larger γ may be used in practice. While we do not guarantee the same results at higher γ, we observe them to empirically hold. The use of variance as a fairness metric is a choice we make for our experiments. However, other metrics may also be considered, as long as agents' contributions can be independently extracted. It is also possible to use multi agent RL techniques like VDN (Sunehag et al., 2018) or QMIX (Rashid et al., 2020) in conjunction with our methods to learn how to divide the fairness reward across agents.

Acknowledgments

This research is partially supported by the National Science Foundation under award 2232055. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the United States government.

References

Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, 114:462–467, 2017.

Frits De Nijs, Erwin Walraven, Mathijs De Weerdt, and Matthijs Spaan. Constrained multiagent Markov decision processes: A taxonomy of problems and algorithms. *Journal of Artificial Intelligence Research*, 70:955–1001, 2021.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the Conference on Innovations in Theoretical Computer Science*, pp. 214–226, 2012.

Adam N. Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Management Science*, 68: 9–26, 2022.

Pratik Gajane, Akrati Saxena, Maryam Tavakol, George Fletcher, and Mykola Pechenizkiy. Survey on fair reinforcement learning: Theory and practice. *arXiv preprint arXiv:2205.10032*, 2022.

Chongming Gao, Kexin Huang, Jiawei Chen, Yuan Zhang, Biao Li, Peng Jiang, Shiqi Wang, Zhong Zhang, and Xiangnan He. Alleviating matthew effect of offline reinforcement learning in interactive recommendation. In *Proceedings of the International Conference on Research and Development in Information Retrieval*, pp. 238–248, 2023.

Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *Proceedings of the Conference on Neural Information Processing Systems*, pp. 3323–3331, 2016.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 2094–2100, 2016.

Jiechuan Jiang and Zongqing Lu. Learning fairness in multi-agent systems. In *Proceedings of the Conference on Neural Information Processing Systems*, 2019.

Amanda R. Kube, Sanmay Das, and Patrick J. Fowler. Allocating interventions based on predicted outcomes: A case study on homelessness services. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 622–629, 2019.

Amanda R. Kube, Sanmay Das, and Patrick J. Fowler. Community-and data-driven homelessness prevention and service delivery: optimizing for equity. *Journal of the American Medical Informatics Association*, 30(6):1032–1041, 2023.

Ashwin Kumar, Yevgeniy Vorobeychik, and William Yeoh. Using simple incentives to improve two-sided fairness in ridesharing systems. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pp. 227–235, 2023.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys*, 54(6):1–35, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Zhiwei Tony Qin, Hongtu Zhu, and Jieping Ye. Reinforcement learning for ridesharing: An extended survey. *Transportation Research Part C: Emerging Technologies*, 144:103852, 2022.

Manish Raghavan, Solon Barocas, Jon Kleinberg, and Karen Levy. Mitigating bias in algorithmic hiring: Evaluating claims and practices. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 469–481, 2020.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

Daniel Rigney. *The Matthew Effect: How Advantage Begets Further Advantage*. Columbia University Press, 2010.

Sanket Shah, Meghna Lowalekar, and Pradeep Varakantham. Neural approximate dynamic programming for on-demand ride-pooling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 507–515, 2020.

Umer Siddique, Paul Weng, and Matthieu Zimmer. Learning fair policies in multi-objective (deep) reinforcement learning with average and discounted rewards. In *Proceedings of the International Conference on Machine Learning*, pp. 8905–8915, 2020.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the Conference on Autonomous Agents and Multiagent Systems*, pp. 2085–2087, 2018.

Kai Wang, Sanket Shah, Haipeng Chen, Andrew Perrault, Finale Doshi-Velez, and Milind Tambe. Learning MDPs from features: Predict-then-optimize for sequential decision making by reinforcement learning. In *Proceedings of the Conference on Neural Information Processing Systems*, pp. 8795–8806, 2021.

Matthieu Zimmer, Claire Glanois, Umer Siddique, and Paul Weng. Learning fair policies in decentralized cooperative multi-agent reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pp. 12967–12978, 2021.

A Theoretical Results

Proposition 1. *As $\beta_{test} \rightarrow 0$, all fair-only models behave in a utility-maximizing manner.*

We state this without proof. It is easy to follow how this holds, as at $\beta = 0$, the fairness model does not play any role in the decision making.

Theorem 1. *Given perfect estimates for utility and fairness, increasing β always improves the one-step fairness gain for SO with $\gamma = 0$.*

Proof. We assume that the utility and fairness estimates are converged and correct, i.e. the estimates of fairness and utility are correct. For the following discussion, assume the environment has evolved over some time t and is at a state s_t. We consider what changes when we change β at this state. Variables used henceforth are conditioned on s_t, wherever reasonable. We make the conditioning on s_t implicit and do not notate it, to make it easier to read.

With $\gamma = 0$, the optimal utility and fairness estimates equal the one-step return, i.e. the change in utility and fairness because of the resulting joint action. Note that these values are not known to the agents prior to the allocation as they depend on the joint actions of all agents, so computing these estimates is not trivial.

Let $U_{tot}(\mathcal{A})$ and $F_{tot}(\mathcal{A})$ be defined as follows, given an allocation $\mathcal{A}$:

$$U_{tot}(\mathcal{A}) = \sum_{i \in \alpha} U(\mathcal{A}_i) \tag{15}$$

$$F_{tot}(\mathcal{A}) = \sum_{i \in \alpha} F(\mathcal{A}_i) \tag{16}$$

We remind the reader that $\mathcal{A}_i$ refers to the action assigned to agent i in the allocation $\mathcal{A}$.

Let $\mathbf{Z}_t$ represent the agent metrics at time t. Further, let $\mathcal{A}^*$ represent the optimal allocation from the ILP with β as the tradeoff weight. Since $\mathcal{A}^*$ is optimal, it follows that for all other possible allocations $\mathcal{A}_o$,

$$U_{tot}(\mathcal{A}^*) + \beta F_{tot}(\mathcal{A}^*) \geq U_{tot}(\mathcal{A}_o) + \beta F_{tot}(\mathcal{A}_o) \tag{17}$$

$$U_{tot}(\mathcal{A}^*) - U_{tot}(\mathcal{A}_o) \geq \beta(F_{tot}(\mathcal{A}_o) - F_{tot}(\mathcal{A}^*)) \tag{18}$$

We are interested in finding what happens to the allocation when we increase β. For $\beta' > \beta$, note that the left side of Eq.18 remains the same. Since utility estimates are not affected by changing β, any other allocation $\mathcal{A}_o$ can only be selected over $\mathcal{A}^*$ if the following condition holds:

$$U_{tot}(\mathcal{A}^*) + \beta' F_{tot}(\mathcal{A}^*) \leq U_{tot}(\mathcal{A}_o) + \beta' F_{tot}(\mathcal{A}_o) \tag{19}$$

$$U_{tot}(\mathcal{A}^*) - U_{tot}(\mathcal{A}_o) \leq \beta'(F_{tot}(\mathcal{A}_o) - F_{tot}(\mathcal{A}^*)) \tag{20}$$

Combining Eqs.18 and 20, we get the following:

$$\beta(F_{tot}(\mathcal{A}_o) - F_{tot}(\mathcal{A}^*)) \leq \beta'(F_{tot}(\mathcal{A}_o) - F_{tot}(\mathcal{A}^*)) \tag{21}$$

$$F_{tot}(\mathcal{A}^*)(\beta' - \beta) \leq F_{tot}(\mathcal{A}_o)(\beta' - \beta) \tag{22}$$

Since $\beta \geq 0$ and $\beta' > \beta$, Eq.22 can only be true if $F_{tot}(\mathcal{A}_o) > F_{tot}(\mathcal{A}_o)$.

Thus, any allocation $\mathcal{A}_o$ that is optimal (and thus selected by the ILP) for $\beta' > \beta$ is guaranteed to have equal or better fairness than the allocation $\mathcal{A}^*$ at β.

□

We also state the corollary to Theorem 1.

Corollary 1. *Given perfect estimates for utility and fairness, decreasing β always improves the one-step utility gain for SO with $\gamma = 0$.*

The proof follows a similar structure to Theorem 1

While we only prove the behavior for $\gamma = 0$, our empirical results show that we can expect similar behavior for long-horizon estimates. For any state, we will select actions that improve fairness in the long run starting from that state as β is increased.

We also show the following useful property

Theorem 2. *For a large enough β, the fairest allocation will be selected with perfect utility and fairness estimators for SO with $\gamma = 0$.*

Proof. Let $\mathcal{A}_f$ denote the allocation with the largest fairness gain

$$\mathcal{A}_f = \operatorname*{argmax}_{\mathcal{A}} F_{tot}(\mathcal{A})$$

For simplicity, let us assume no two allocations have the same $F_{tot}(\mathcal{A})$. For any other allocation $\mathcal{A}_o$, we have

$$F_{tot}(\mathcal{A}_f) > F_{tot}(\mathcal{A}_o) \tag{23}$$

Then, $\mathcal{A}_f$ will be optimal and selected by the ILP if the following condition holds.

$$U_{tot}(\mathcal{A}_f) + \beta_f F_{tot}(\mathcal{A}_f) \geq U_{tot}(\mathcal{A}_o) + \beta_f F_{tot}(\mathcal{A}_o) \tag{24}$$

$$\beta_f \geq \frac{U_{tot}(\mathcal{A}_o) - U_{tot}(\mathcal{A}_f)}{F_{tot}(\mathcal{A}_f) - F_{tot}(\mathcal{A}_o)} \tag{25}$$

We can compute an upper bound for β_f by considering the range of values that U_{tot} and F_{tot} can take. Let $U_{max} = \max_{\mathcal{A}} U_{tot}(\mathcal{A})$, and $F_{max} = \max_{\mathcal{A}, \mathcal{A} \neq \mathcal{A}_f} F_{tot}(\mathcal{A})$.

Then, we have the following:

$$\beta_f \geq \frac{U_{tot}(\mathcal{A}_o) - U_{tot}(\mathcal{A}_f)}{F_{tot}(\mathcal{A}_f) - F_{tot}(\mathcal{A}_o)} \tag{26}$$

$$\leq \frac{U_{max} - U_{tot}(\mathcal{A}_f)}{F_{tot}(\mathcal{A}_f) - F_{tot}(\mathcal{A}_o)} \tag{27}$$

$$\leq \frac{U_{max} - U_{tot}(\mathcal{A}_f)}{F_{tot}(\mathcal{A}_f) - F_{max}} = \beta_f^u \tag{28}$$

Eq.28 gives us an upper bound for β_f. Thus, for all $\beta > \beta_f^u$, $\mathcal{A}_f$ will be the optimal allocation. □

Corollary 2. *For a small enough β, the most utilitarian allocation will be selected with perfect utility and fairness estimators for SO with $\gamma = 0$*

The proof follows a similar structure to the proof for the previous theorem.

B Environment Details

Here, we provide further details about the environments for reproducibility. We will also make the code available upon acceptance.

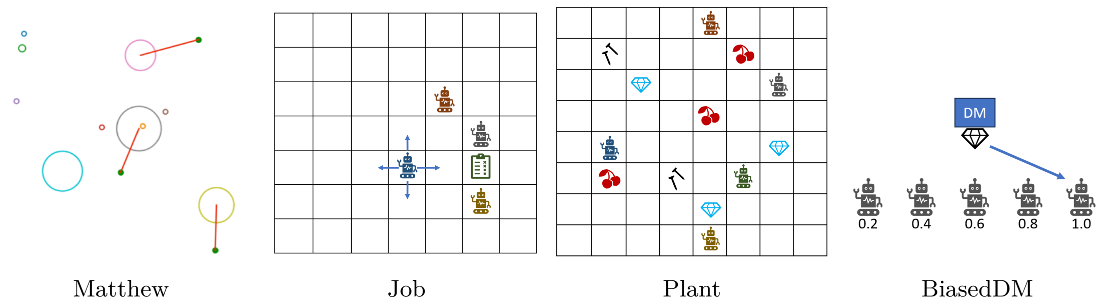


Figure 4: Illustration of all four environments

B.1 Matthew

One episode for this environment lasts 200 steps. At each step, 10 agents and 3 resources are available on the map. Agent speeds grow proportionally to their size, and a size ceiling exists to prevent agents from growing too large for the environment bounds. Agent and resource positions are 2-D coordinates in $[0, 1]$. At the beginning of each episode, agent and resource positions are randomly initialized. To show the matthew effect, 4 agents are initialized to have a larger initial size than other agents. This allows them to reach resources faster. Agents receive a unit reward when they collect a resource, in addition to a small increase in size and speed. Resources allocated to agents are reserved, and other agents cannot pick them up. A new resource only spawns when an agent reaches its allocated resource, not when it is allocated.

B.2 Job

One episode for this environment lasts 100 steps. There are 4 agents on a 7×7 grid, and agents can move in any cardinal direction or stay in their location. Agents cannot occupy the same location as any other agent, and they receive a unit reward when occupying the resource location. Attempting to move out of the grid results in a no-op. The agents' initial locations are randomly generated each episode. The job's location is also randomly selected, but it is not allowed to be on the border cells of the grid, and its position remains fixed for the entire episode. The objective in this environment is for agents to learn to share the job instead of occupying it alone.

B.3 Plant

One episode for this environment lasts 200 steps. There are 5 agents on a 8×8 grid, where agents can move in cardinal directions. The grid also contains 8 resources, which can be of three different types. Each agent gets a reward when they construct a 'unit'. Each agent has a requirement of a set of resources it must collect so that it can construct this unit. The requirements are:

$$\{(2, 1, 0), (1, 0, 1), (1, 0, 0), (1, 3, 0), (0, 1, 2)\}$$

For example, agent 1 requires two resources of type 1 and one resource of type 2 following which it can get a reward. Some agents are given easier requirements to fulfill, which creates a bias in the number of units agents produce. The resources and agent locations are randomly initialized at the beginning of each episode. The allocation follows a similar process to the Matthew environment, where actions are allocations of agents to resources, and other agents cannot pick up resources already allocated to other agents. When a resource is picked up, another resource of the same type appears in a random location on the map.

B.4 BiasedDM

One episode for this environment lasts 100 steps. At each time step, the decision maker allocates one resource to one of five agents. The utility of assigning the resource to each agent is different for the decision maker. In other environments, fairness is computed as the variance over the accumulated rewards for each agent. In this environment, however, fairness is computed over the resource rate, which is the fraction of steps in which an agent received the resource ($z_i \in [0, 1]$).

$$z_i = \frac{\text{Num. resources}}{time} \tag{29}$$

This also results in much smaller variances, thus our hyperparameter search for this domain explores a much larger range of β values.

C Model architecture and training

All our models use a learning rate of 0.0003 with the Adam optimizer. For all environments except BiasedDM, we train for 1000 episodes, and run validation every 50 steps for model selection. For BiasedDM, we train for 200 episodes, and validate every 20 episodes.

The neural network architecture for all models is the same, with two fully connected hidden layers, of dimension 20, with ReLU activations. The output (1-dimensional) does not have any activation function. We implement our networks using pytorch.

We use a replay buffer of size 250000, where one experience is a joint transition across all agents. During training, we sample experiences from the replay buffer, and for each experience, we evaluate actions for all agents using the current online network, solving the ILP to get the best joint action. Then, we score the post-decision state for each agent using the target network, and compute the MSE loss between the target value and value estimates of the selected action from the online network.

We ran all our experiments on a university compute cluster, with each experiment running on a single CPU node with 8GB RAM. Experiment runtime varied with environment choice. Training a single model with one β value took between 30 minutes (BiasedDM) and 2 hours (Matthew). Evaluation, as for the generalization experiments, was performed on a 2019 MacBook Pro, where one episode took 2-5 seconds to run, and we bootstrap over 5 runs.

D Extended Results

Here we provide additional results for our methods, including providing confidence intervals for our main results. We provide confidence intervals for system utility and fairness separately as we vary β, as plotting this on a pareto plot (as in the main results) would be hard to read (Figure 5). We also provide the generalization results for varying β_{test} for both Split (Figure 6) and Fair Only (Figure 7) models.

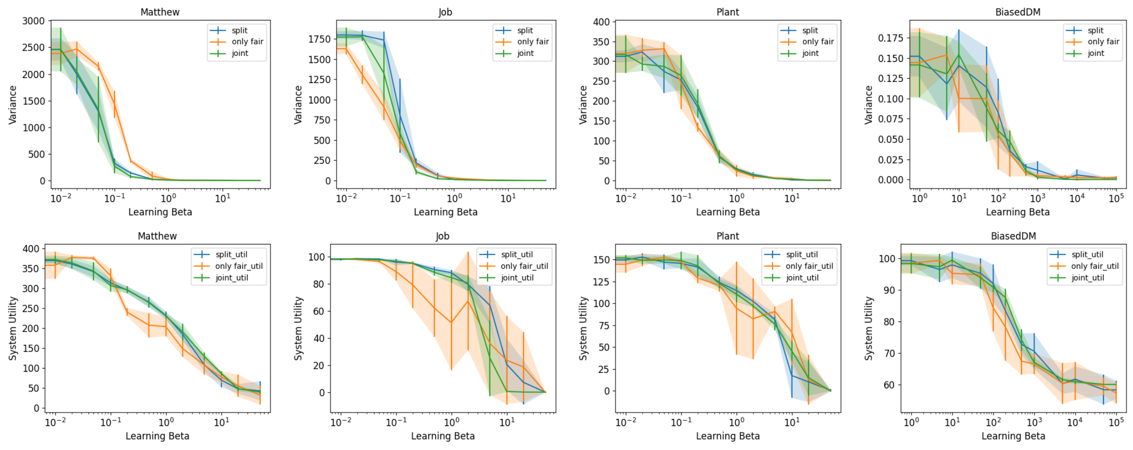


Figure 5: Effect of changing β on variance (top row) and utility (bottom row) for all three methods on all four environments. The shaded area shows the 1-σ error bar. We observe that the performance of FO has large variation for intermediate β values.

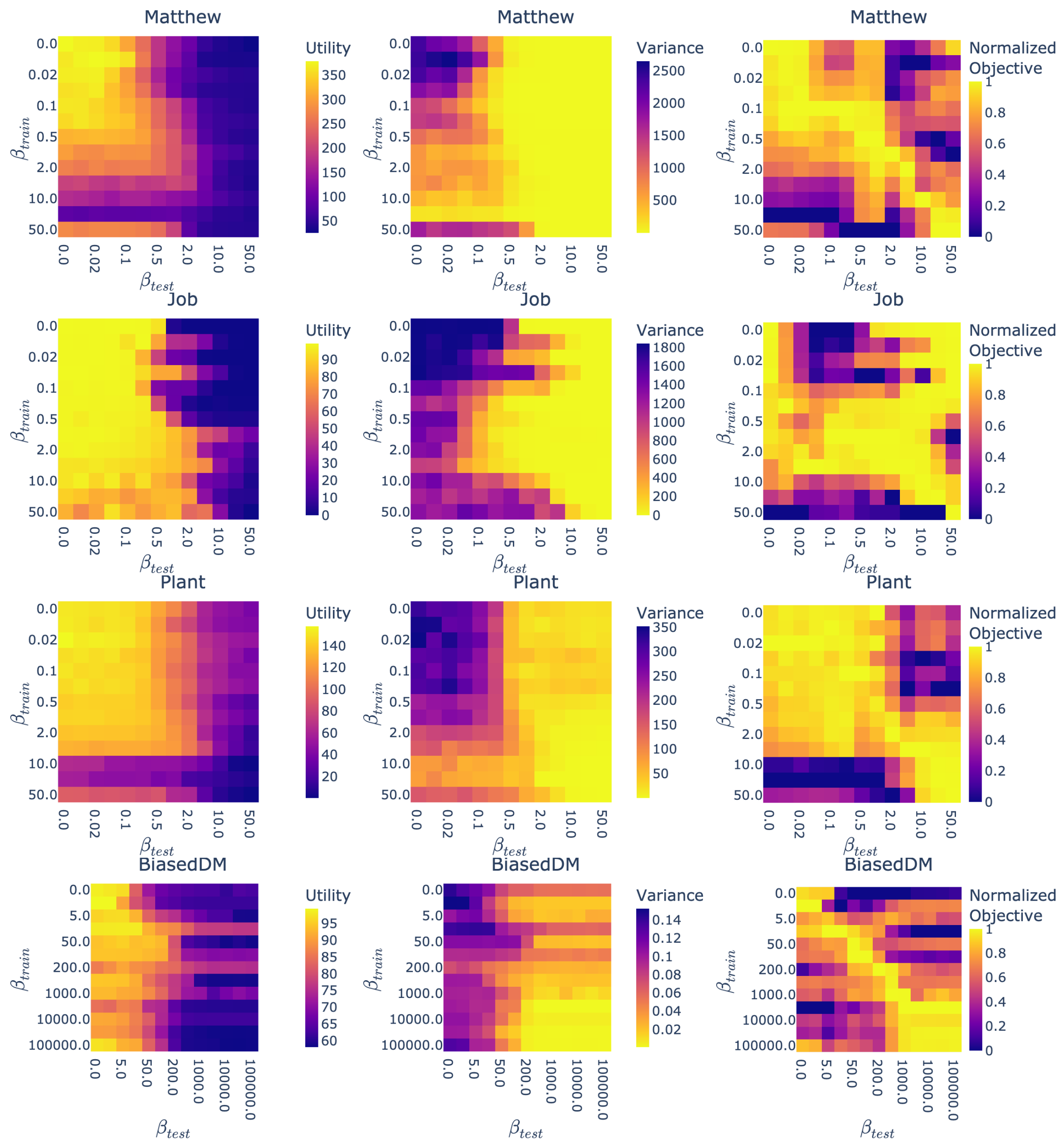


Figure 6: Evaluation of Split Optimization models trained on β_{train} and evaluated on β_{test} for all environments. Brighter colors indicate better outcomes. The "Normalized Objective" plot scales the values in each column of the heatmap between 0 and 1.

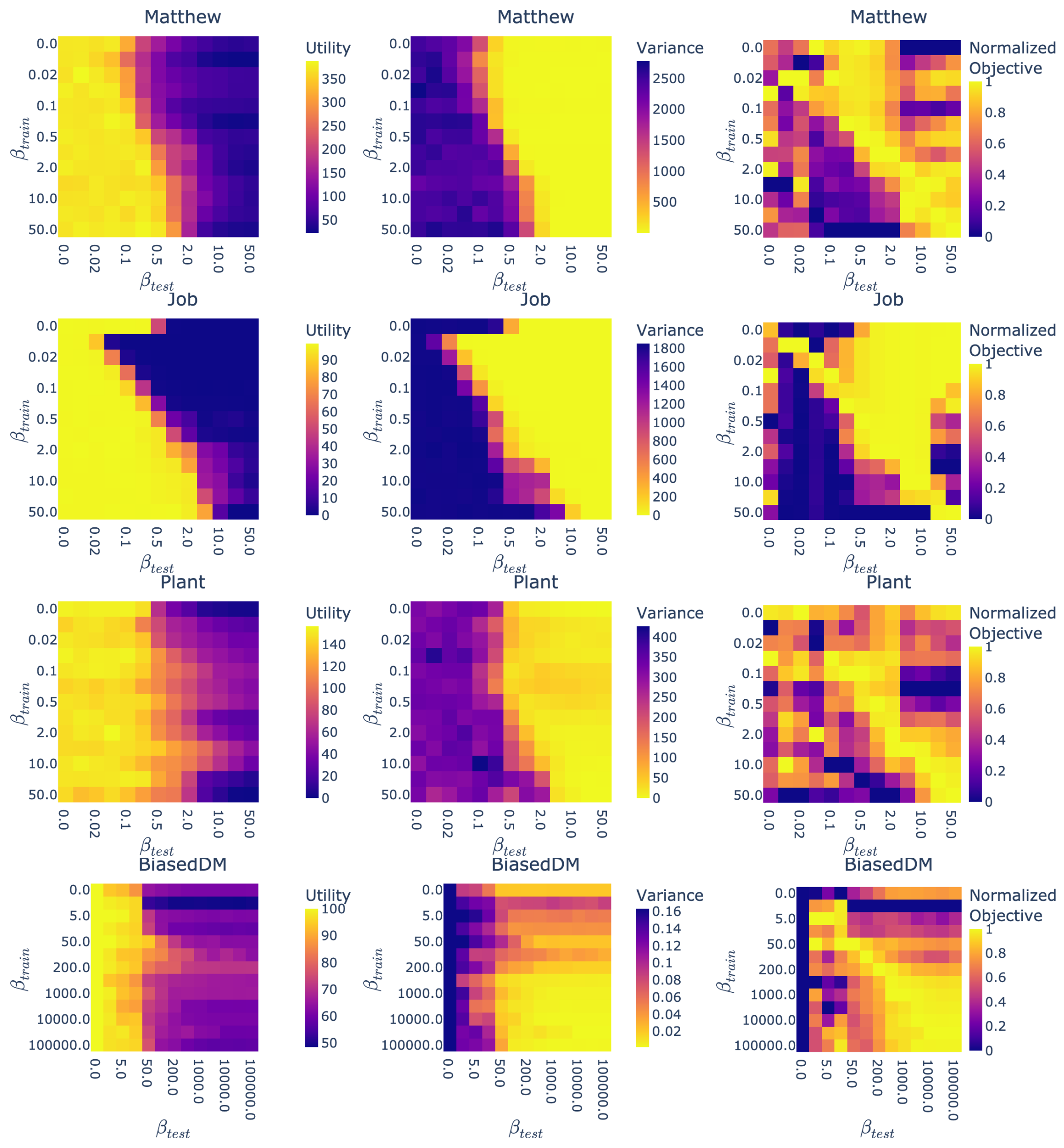


Figure 7: Evaluation of Fair Only models trained on β_{train} and evaluated on β_{test} for all environments. Brighter colors indicate better outcomes. The "Normalized Objective" plot scales the values in each column of the heatmap between 0 and 1.