Fair Deep Reinforcement Learning with Generalized Gini Welfare Functions

Guanbao Yu
UM-SJTU Joint Institute, Shanghai
Jiao Tong University
Shanghai, China
gbyu66@sjtu.edu.cn

Umer Siddique
University of Texas
San Antonio, USA
umersiddique297@gmail.com

Paul Weng
UM-SJTU Joint Institute, Shanghai
Jiao Tong University
Shanghai, China
paul.weng@sjtu.edu.cn

ABSTRACT

Learning fair policies in reinforcement learning (RL) is important when the RL agent's actions may impact many users. In this paper, we investigate a generalization of this problem where equity is still desired, but some users may be entitled to preferential treatment. We formalize this more sophisticated fair optimization problem in deep RL, provide some theoretical discussion of its difficulties, and explain how existing deep RL algorithms can be adapted to tackle it. Our algorithmic innovations notably include a state-augmented DQN-based method for learning stochastic policies, which also applies to the usual fair optimization setting without any preferential treatment. We empirically validate our propositions and analyze the experimental results on several application domains.

KEYWORDS

Deep reinforcement learning, Fair Optimization, Multi-objective

1 INTRODUCTION

In this paper, we consider adaptive learning agents based on deep reinforcement learning (RL). When they are deployed in real applications (e.g., traffic lights, software-defined networking, data centers), they may interact and impact many users. Hence, for these systems to be accepted by end-users when they are in operation, fairness needs to be taken into account in their design.

Fairness is rooted in the principle of "equal treatment of equals", which informally speaking means that individuals with similar characteristics should be treated in a similar way. Previous work [10, 43] in learning fair policies in RL focuses on such notion with the additional assumption that all individuals are equal, which may not be suitable for all applications. For instance, it is customary for service providers (in e.g., software-defined networking, data centers) to provide different levels of QoS (quality of service) to different user tiers. In such cases, although the principle of "equal treatment of equals" is still a desired objective, higher-paying users should arguably be entitled to higher priority or better services.

In our work, we relax the assumption of equal individuals and consider the more general case where different users may have different rights. Our goal is to investigate this more sophisticated fairness problem in the context of deep RL, where efficient policies should be learned such that while some users may receive preferential treatment, users with similar rights are fairly treated.

Contributions. We formalize this novel problem in deep RL as a fair optimization problem (Section 4.2). We discuss the theoretical aspects and difficulties of this problem (Section 4.3). Based on this discussion, we propose several adaptations of deep RL algorithms to solve this problem (Section 5). Notably, we design a novel state-augmented DQN-based method for learning fair stochastic policies. Finally, we experimentally validate our propositions (Section 6).

Proc. of the Adaptive and Learning Agents Workshop (ALA 2023), Cruz, Hayes, Wang, Yates (eds.), May 29-30, 2023, London, UK, https://alaworkshop2023.github.io/. 2023.

2 RELATED WORK

Due to the realization of the tremendous impact that artificial intelligence (AI) and machine techniques can have on our lives, fairness has recently become an important and active research direction [1, 7, 12, 16, 28, 41, 44, 49, 51, 52]. Although various definitions of fairness have been considered in AI, e.g., proportionality [3, 47] or envy-freeness [11] and its multiple variants (e.g., [4, 9]), the majority of this literature in machine learning focuses on the impartiality aspect of fairness: "equal treatment of equals". Proposed methods in this direction typically rely on a constraint-based or penalty-based formulation in order to control bias at the individual or group level. In contrast, our work is based on studies in distributive justice [5, 26, 38]. We aim at optimizing a social welfare function that encodes impartiality, but also equity and efficiency (see Section 3.4 for more details). This principled approach has also been recently advocated in several recent papers [14, 18, 46, 50] and applied in various machine learning tasks, such as sequential decision-making, which we discuss below, but also ranking [15] for instance.

In mathematical optimization, such an approach is called fair optimization [33]. Many continuous and combinatorial optimization problems in various application domains [2, 31, 32, 34, 42] have been extended to optimize for fairness. In this direction, the closest work [34] regards fair optimization in Markov decision processes. However, the methods proposed in this direction typically assume that the model is known and therefore, they do not require learning.

Fairness in RL starts to receive more attention. Different directions have been studied, e.g., fairness constraint to reduce discrimination [49], fairness with respect to state visitation [17, 19], the usual case of fairness with respect to agents [20], or the more general case of fairness with respect to users [10, 23, 43, 53]. This last direction can be understood as an extension of fair optimization to (deep) RL. Our work follows this principled approach, but investigates a more general setting. While previous work assumes all users to be equal, we relax this assumption.

State augmentation (used in our DQN variants) has been exploited in various previous work, e.g., in MDPs [21] or more recently in safe RL [45], risk-sensitive RL [13], RL with delays [30], and partially-observable path planning [29]. However, to the best

of our knowledge, this technique has not been applied in fair optimization. Moreover, our technique to learn stochastic policies in DQN is also novel.

3 BACKGROUND

We first recall the Markov decision process (MDP) model and RL, then present the multi-objective extension of MDP. We also provide an overview of several deep RL algorithms. Finally, we review the social welfare functions (SWFs) that we used to encode fairness in deep RL.

Notations. Both the matrices and vectors are written in bold. For any vector $\boldsymbol{u} \in \mathbb{R}^D$, $\boldsymbol{u}^{\uparrow}$ corresponds to the vector with the components of vector $\boldsymbol{u}$ sorted in an increasing order (i.e., $\boldsymbol{u}_1^{\uparrow} \leq \ldots \leq \boldsymbol{u}_D^{\uparrow}$). For any integer $D > 0$, the $D-1$ simplex is denoted by $\Delta_D = \{\boldsymbol{w} \in \mathbb{R}^D \mid \sum_i \boldsymbol{w}_i = 1 \text{ and } \boldsymbol{w}_i \geq 0, i = 1, \ldots, D\}$. We denote $\mathbb{S}_D$ the symmetric group of degree D (i.e., set of permutations over $\{1, \ldots, D\}$). For any permutation $\sigma \in \mathbb{S}_D$ and vector $\boldsymbol{u} \in \mathbb{R}^D$, vector $\boldsymbol{u}_\sigma$ denotes $(\boldsymbol{u}_{\sigma(1)}, \ldots, \boldsymbol{u}_{\sigma(D)})$.

3.1 Markov Decision Process and RL

A Markov Decision Process (MDP) [37] model is characterized by its set of states $\mathcal{S}$, set of actions $\mathcal{A}$, a transition model P which specifies the probability of reaching next state s' by taking action a in state s, and a reward function R indicating the immediate reward of performing action a in state s. This model also includes the discount factor $\gamma \in [0, 1)$, and the probability distribution over the initial states d_0.

A *policy* π in an MDP model provides guidance on which action to take in any state s. It can be deterministic if $\pi(s) = a$ or stochastic if $\pi(a \mid s) = Pr(a \mid s)$. Note that deterministic policies are special cases of stochastic ones. For a policy π, we denote P_π (resp. r_π) the transition (resp. reward) function induced by π, i.e., $P_\pi(s, s') = P(s, \pi(s), s')$ (resp. $r_\pi(s) = r_\pi(s, \pi(s))$). The usual goal in MDP is to learn a policy π that maximizes the expected discounted reward, i.e., $\mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t\right]$.

Formally, the *(state) value function* $v_\pi : \mathcal{S} \to \mathbb{R}$ of a policy π from an initial state s is defined by:

$$v_\pi(s) = \mathbb{E}_{P,\pi}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \mid s\right], \tag{1}$$

where $\mathbb{E}_{P,\pi}$ is the expectation taken with respect to transition function P and policy π, and r_t is the random variable that represents the reward obtained at time step t. The value function v_π provides the expected discounted reward one can get by following the corresponding policy π from state s. Similarly, the *action-value function* $Q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is given by:

$$Q_\pi(s, a) = \mathbb{E}_{P,\pi}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \mid s, a\right]. \tag{2}$$

Formally, both the MDP and RL attempt to address the following optimization problem: $\text{argmax}_\pi \sum_{s \in \mathcal{S}} d_0(s) v_\pi(s)$, where d_0 is the initial state distribution and v_π is the value function approximated by following the current policy π. A solution to this problem is an *optimal* policy, which is denoted by π^*.

3.2 Multiobjective Markov Decision Process

We formulate the novel fair optimization problem as a multiobjective MDP (MOMDP), where each objective corresponds to the individual utility of a user in our setting. Therefore, the rewards in MOMDPs are vectors instead of scalars. The reward function of a MOMDP can be formalized as $\boldsymbol{r}(s, a) \in \mathbb{R}^D$ where D is the number of objectives (users).

All the previous definitions in MDP can be naturally extended to MOMDP. For example, the value function in (1) now becomes:

$$\boldsymbol{v}_\pi(s) = \mathbb{E}_{P,\pi}\left[\sum_{t=1}^{\infty} \gamma^{t-1} \boldsymbol{r}_t \mid s\right], \tag{3}$$

where $\boldsymbol{r}_t \in \mathbb{R}^D$ is the vector reward obtained at time step t and all the operations (addition, product) are component-wise.

3.3 Deep RL

Deep RL is the study of RL using neural networks as function approximators. They are needed to tackle large-scale RL problems, where the state and/or action spaces become large or continuous. With parametric function approximation such as neural networks, a function f is approximated by $\hat{f}_{\boldsymbol{\theta}}$ where $\boldsymbol{\theta}$ denotes the parameters of the parametric function, which can be learned during training. In RL, both value functions or policies can be approximated.

Deep Q-Network (DQN) [25] is an example of deep RL algorithm where the optimal Q function is approximated by a neural network with parameter $\boldsymbol{\theta}$. This Q-network takes a state s as input and outputs an estimated $\hat{Q}_{\boldsymbol{\theta}}(s, a)$ for all actions. It is trained to minimize the following L_2 loss for a sampled transition (s, a, r, s'):

$$(r + \gamma \hat{Q}_{\boldsymbol{\theta}'}(s', a^*) - \hat{Q}_{\boldsymbol{\theta}}(s, a))^2$$

where $a^* = \text{argmax}_{a' \in \mathcal{A}} \left(r + \gamma \hat{Q}_{\boldsymbol{\theta}'}(s', a')\right)$ and $\boldsymbol{\theta}'$ represents the parameters of the target Q-network which promotes more stable training. The transitions (s, a, r, s') are sampled from a replay buffer storing experiences generated from online interactions with the environment. The term $r + \gamma \hat{Q}_{\boldsymbol{\theta}'}(s', a^*)$ is called *target Q-value*.

Policy gradient methods constitute another approach for solving RL problems. In contrast to value-based methods like DQN, policy gradient methods explicitly optimize the desired objective function in a parameterized policy space, with the goal of finding a policy $\pi_{\boldsymbol{\theta}}(a \mid s)$ ($\boldsymbol{\theta}$ being the policy parameters) that maximizes the expected sum of reward. In policy gradient methods, the objective function $J(\boldsymbol{\theta})$ can be formally defined as:

$$J(\boldsymbol{\theta}) = \sum_{s \in S} d_{\pi_{\boldsymbol{\theta}}}(s) V_{\pi_{\boldsymbol{\theta}}}(s) = \sum_{s \in S} d_{\pi_{\boldsymbol{\theta}}}(s) \sum_{a \in A} \pi_{\boldsymbol{\theta}}(a \mid s) Q_{\pi_{\boldsymbol{\theta}}}(s, a), \tag{4}$$

where $d_{\pi_{\boldsymbol{\theta}}}(s)$ is the stationary state distribution under policy $\pi_{\boldsymbol{\theta}}$.

Parameter $\boldsymbol{\theta}$ can be learned using gradient ascent by following the update direction given by the *Policy Gradient Theorem* [48]:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{s \sim \boldsymbol{d}_\pi, a \sim \pi_{\boldsymbol{\theta}}(\cdot|s)} [Q_{\pi_{\boldsymbol{\theta}}}(s, a) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)]. \tag{5}$$

where the Q-value function $Q_{\pi_{\boldsymbol{\theta}}}(s, a)$ can be estimated using Monte-Carlo or temporal difference methods.

3.4 Fairness

In this paper, an optimal fair solution is required to satisfy three properties [43]: efficiency, equity, and impartiality. The efficiency

property states that a solution should be Pareto-optimal. This is a natural property because selecting a Pareto-dominated solution would be irrational. The equity property is based on the *Pigou-Dalton principle* [27], which states that transferring utility from a better-off user to a worse-off user results in a fairer solution. This principle establishes the foundation of fairness by distributing equal wealth among different users, which is a critical component in our definition of fairness. The impartiality property corresponds to the *"equal treatment of equals"* principle. This principle served as the foundation for previous works that assume all users are equal. However, in our work, we relax this assumption and consider a more general case in which some users may be given preference over others.

We rely on *social welfare functions (SWFs)* to formalize these three properties as an objective function. An SWF evaluates how good a solution is for all users by aggregating all users' utilities. In this paper, we only discuss those SWFs that satisfy our notion of fairness and refer to them as *fair SWFs*. One notable group of fair SWFs in the literature is the *generalized Gini social welfare function* (GGF), which is defined as follows:

$$GGF_{\boldsymbol{w}}(\boldsymbol{u}) = \sum_{i=1}^{D} \boldsymbol{w}_i \boldsymbol{u}_i^{\uparrow}, \tag{6}$$

where $\boldsymbol{u} \in \mathbb{R}^D$ and $\boldsymbol{w} \in \Delta_D$ is a fixed positive weight vector whose components are strictly decreasing (i.e., $\boldsymbol{w}_1 > \ldots > \boldsymbol{w}_D > 0$). Intuitively, by assigning larger weights on smaller utility values, GGF will yield larger scores when the utility distribution becomes more balanced while keeping the total utility constant.

GGF satisfies all three of the above-mentioned properties. Since GGF is a strictly increasing function with positive weights, it implies that it is monotonic in terms of Pareto-dominance and thus meets the efficiency property. GGF also satisfies the equity property because it is a strictly Schur-concave function, which implies that it is monotonic with respect to Pigou-Dalton transfers. Finally, because the components of GGF are symmetric (i.e., independent of the order of their arguments), it satisfies the impartiality property.

Despite the fact that GGF is a simple yet effective SWF for encoding fairness, it has some limitations. For instance, the symmetry of GGF entails that it only applies to cases where all users are equal. However, in many real-world applications, some objectives/users may be preferred. For instance, as discussed in the introduction, the service providers controlled by autonomous systems have to take different user tiers into account. For such systems, a fair SWF that can encode preferences over objectives is required, which we will explain in the following section.

4 FAIR OPTIMIZATION WITH PREFERENTIAL TREATMENT

In this section, we first extend GGF to a generalized fair SWF that can encode preferential treatment, which we call *generalized GGF* (G^3F). Based on G^3F, we then formulate this novel fair optimization problem in deep RL. Finally, we explain the difficulties of solving the above problem and present some theoretical discussion.

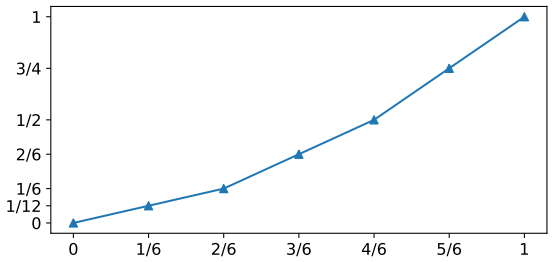


Figure 1: Linear interpolation graph for different x values

4.1 G^3F

G^3F extends GGF by introducing an additional weight $\boldsymbol{p}$ to encode preferential treatment. The weight $\boldsymbol{p}$ is also called *importance weight*. Formally, let $\boldsymbol{p} \in \Delta_D$ and $\boldsymbol{w} \in \Delta_D$ be two fixed weighting vectors, the G^3F is defined as follows:

$$G^3F_{\boldsymbol{p},\boldsymbol{w}}(\boldsymbol{u}) = \sum_i \boldsymbol{\omega}_i \boldsymbol{u}_i^{\uparrow}, \tag{7}$$

where $\boldsymbol{u} \in \mathbb{R}^D$ and the weight $\boldsymbol{\omega}_i$ is defined as:

$$\boldsymbol{\omega}_i = w^*\left(\sum_{k=1}^{i} \boldsymbol{p}_{\sigma(k)}\right) - w^*\left(\sum_{k=1}^{i-1} \boldsymbol{p}_{\sigma(k)}\right), \tag{8}$$

with w^* being a monotone increasing function that linearly interpolates the points $(i/D, \sum_{k=1}^{i} \boldsymbol{w}_k)$ together with the point $(0, 0)$, and σ is the permutation sorting the components of vector $\boldsymbol{u}$ in increasing order, i.e., $\boldsymbol{u}_{\sigma(i)} = \boldsymbol{u}_i^{\uparrow}$ for all i.

Intuitively, such preferential treatment can be enforced via *user duplication* [6], which states that if a user is more important, s/he should be counted more times (via importance weight) than other users. Since this weight is often normalized, formally, if each user i receives some fractional entitlement $\boldsymbol{p}_i$ (i.e., importance weights), when two users are equally important, they would receive equal weights. In contrast, if a user is entitled to a preferential treatment, s/he would consequently receive a larger share of the total importance weight. The exact choice of $\boldsymbol{p}$ therefore depends on the specific problem one wants to solve.

Recall that G^3F in (7) is defined with positive decreasing weights $\boldsymbol{w}$, it therefore satisfies efficiency, equity, and impartiality, but without assuming that all users are equal. Obviously, G^3F will reduce to GGF when $\boldsymbol{p}$ follows a uniform distribution. For a better illustration on why G^3F is a suitable choice in our setting, we consider the following example.

Example 4.1. Assume given an instance of $G^3F_{\boldsymbol{p},\boldsymbol{w}}$ where $\boldsymbol{w}$ is chosen as $(3/6, 2/6, 1/6)$ and $\boldsymbol{p}$ is set to $(4/6, 1/6, 1/6)$. By applying linear interpolation at the key points: $w^*(0) = 0$, $w^*(1/3) = 1/6$, $w^*(2/3) = 1/2$, $w^*(1) = 1$, we can obtain the complete values (see Figure 1). The following cases show that G^3F satisfies the three properties of a fair solution.

In the first case, We consider two vectors $\boldsymbol{u} = (10, 5, 15)$, and $\boldsymbol{u}' = (12, 5, 15)$. By efficiency, $\boldsymbol{u}'$ should be preferred to $\boldsymbol{u}$, which is true by comparing the corresponding aggregation values:

$$G^3F_{\boldsymbol{p},\boldsymbol{w}}(\boldsymbol{u}) = 9.17, G^3F_{\boldsymbol{p},\boldsymbol{w}}(\boldsymbol{u}') = 10.50.$$

Let $\boldsymbol{u} = (10, 5, 15)$, and $\boldsymbol{u}' = (10, 12, 8)$ in the second case. By equity, $\boldsymbol{u}'$ should be preferred to $\boldsymbol{u}$ since the last two objectives are equally important and $\boldsymbol{u}'$ is more balanced over them. Indeed, we have:

$$\mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}(\boldsymbol{u}) = 9.17, \mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}(\boldsymbol{u}') = 9.67.$$

Therefore the equity property holds.

In the lase case, we consider $\boldsymbol{u} = (10, 5, 15)$, and $\boldsymbol{u}' = (10, 15, 5)$. And we can obtain:

$$\mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}(\boldsymbol{u}) = \mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}(\boldsymbol{u}') = 9.17.$$

Thus the two solutions are equivalent, which verifies the impartiality property.

4.2 Problem Statement

By integrating G^{3}F with MOMDPs, we can now formally formulate this fair optimization problem with preferential treatment investigated in our paper, which is the problem of determining a policy that generates a fair distribution of rewards subject to the preference weighting vector. Since we focus on deep RL, we directly write this problem with parametrized policy $\pi_{\boldsymbol{\theta}}$:

$$\underset{\pi_{\boldsymbol{\theta}}}{\operatorname{argmax}}\, \mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}(\boldsymbol{J}(\pi_{\boldsymbol{\theta}})), \tag{9}$$

where $\boldsymbol{J}(\pi_{\boldsymbol{\theta}})$ corresponds to the vectorial version of the standard RL objective. A solution to this problem is called G^{3}F-*fair* policy or simply *fair* policy if the context is clear. Note that both $\boldsymbol{p}$ and $\boldsymbol{w}$ are fixed and depend notably on the problem domain, its context, and what the system designer wants to achieve. These weights are therefore part of the problem description.

While the usual approaches in MOMDPs aim to find the set of Pareto optimal solutions (or an approximation), the goal of our problem is to directly learn the Pareto-optimal G^{3}F-fair policy. In addition, instead of applying G^{3}F on the immediate rewards, our formulation applies G^{3}F on the cumulative rewards over trajectories to reach more equitable reward distribution, since it allows compensation over time and expectation in this way.

4.3 Difficulties

Similarly to GGF optimization in RL [43], several challenges exist for solving Problem (9): (i) G^{3}F is a non-linear function, which makes the problem harder to solve than standard RL. However, interestingly, G^{3}F is a concave function (see Section 4.3.1), which suggests that (9) may still retain some nice properties. (ii) fair solutions may depend on initial states. (iii) stochastic policies may dominate deterministic policies when taking fairness into account.

For GGF, Siddique et al. [43] also discuss those points for the average reward criterion, and in addition, introduce an approximation bound in terms of average reward between the policy optimal for the discounted reward and that for the optimal reward. Those results can be extended to G^{3}F, but to keep the exposition simple, we do not present them in this paper.

4.3.1 Concavity Analysis. Although Ogryczak and Śliwiński [36] have proved the concavity of G^{3}F, here we provide another straightforward proof as an alternative.

Lemma 4.2. *For any $\boldsymbol{p} \in \Delta_D$, for any $\boldsymbol{w} \in \Delta_D$ such that its components are decreasing, function $\mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}$ is concave.*

Proof. We prove that $\mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}$ is a Choquet integral with respect to a super-modular capacity. By [22], such integrals are concave[1]. □

The concavity of G^{3}F implies that the optimization problem (9) has some nice properties. For instance, with a linear approximation scheme, the overall problem would be a convex optimization problem (i.e., any local optimum would be global). In deep RL, the overall problem is not convex anymore, but from the point of view of the last layer of a neural network (which is usually linear, e.g., in DQN), the optimization problem is still convex. This suggests that the overall problem is relatively well-behaved, and provides guidance on our algorithm design (see Section 5).

4.3.2 State-dependent Optimality. As an extension of the problem investigated by [43], similarly to GGF-fair policy, an G^{3}F-fair policy may depend on the initial states or more generally, on the distribution of initial states. Related to this point, because of the non-linearity of G^{3}F, the Bellman principle of optimality does not hold anymore and dynamic programming can not be directly applied for finding a fair optimal policy.

4.3.3 Optimality of stochastic policies. It is known that an optimal deterministic policy exists in the single-objective MDP setting. However, when taking fairness into account in the MOMDP setting, learning a policy only from the set of deterministic policies may not be optimal [7]. For instance, given a MOMDP with two objectives, mixing a policy optimal for the first objective with a policy optimal for the second objective can lead to a better trade-off between the two objectives (i.e., fairer solution).

5 PROPOSED ALGORITHMS

In this section, we explain how to integrate G^{3}F with several existing RL algorithms (DQN, A2C, and PPO) for solving Problem (9). Notably, we introduce a novel state-augmented DQN-based method for learning stochastic policies.

5.1 Value-based Methods

Value-based RL methods aim to estimate the optimal action-value function, namely Q_{π^*}. DQN [25] is one typical value-based deep RL method. We discuss next its extension to G^{3}F.

G^{3}F-***DQN***. Following [43], we modify the output of the deep Q-network to take values in $\mathbb{R}^{|\mathcal{A}|\times D}$ instead of $\mathbb{R}^{|\mathcal{A}|}$. The target Q-value is changed to:

$$\hat{\boldsymbol{Q}}_{\boldsymbol{\theta}}(s, a) = \boldsymbol{r} + \gamma \hat{\boldsymbol{Q}}_{\boldsymbol{\theta}'}(s', a^*),$$

where $a^* = \operatorname{argmax}_{a' \in \mathcal{A}} \mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}\big(\boldsymbol{r} + \gamma \hat{\boldsymbol{Q}}_{\boldsymbol{\theta}'}(s', a')\big)$. The best next action is chosen such that the immediate reward plus discounted future rewards (both vectorial) is fair. For execution in a state s, an action in $\operatorname{argmax}_{a \in \mathcal{A}} \mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}\big(\hat{\boldsymbol{Q}}_{\boldsymbol{\theta}}(s, a)\big)$ is chosen. This adapted version of DQN is called G^{3}F-*DQN*.

It is similar to GGF-DQN proposed by Siddique et al. [43]. Here, one may notice that G^{3}F-DQN implicitly optimizes the lower bound[2]

[1] Please refer to the full version of this paper for the detailed proof.

[2] $\mathbb{E}_{s'}\left[\mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}\big(\boldsymbol{r} + \gamma \hat{\boldsymbol{Q}}_{\boldsymbol{\theta}'}(s', a^*_s)\big)\right]$ is a lower bound of $\mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}\big(\mathbb{E}_{s'}\left[\boldsymbol{r} + \gamma \hat{\boldsymbol{Q}}_{\boldsymbol{\theta}'}(s', a^*_s)\right]\big)$ by Jensen inequality since $\mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}$ is concave. Notation a^*_s is to emphasize its dependence on s.

$\mathbb{E}_{s'}\left[\mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}\left(\boldsymbol{r}+\gamma\,\hat{\boldsymbol{Q}}_{\boldsymbol{\theta}'}(s',\,a_s^*)\right)\right]$ instead of $\mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}\left(\mathbb{E}_{s'}\left[\boldsymbol{r}+\gamma\hat{\boldsymbol{Q}}_{\boldsymbol{\theta}'}(s',\,a_s^*)\right]\right)$, which would be a better approximation of the objective function of (9). For this reason, one may not expect a very good performance from G^{3}F-DQN. Next, we propose two other novel extensions of DQN that can achieve better performance.

G^{3}F-***CDQN***. Recall that an optimal fair policy may depend on initial states (Section 4.3.2), and that in G^{3}F-DQN, the learned policy is both deterministic and Markov, which is not sufficient to achieve fairness in an effective way. While still aiming for a deterministic policy here, a natural approach to address the other two points is state augmentation. Indeed, if the agent can base its decisions on both past accumulated reward and usual state information, the agent may be able to achieve a higher level of fairness. Intuitively, such additional information enables the agent to base its decisions on past accumulated reward, which can help correct past inequities.

Consequently, we first augment an original state s_t as follows:

$$\bar{s}_t = \left(s_t, \frac{1}{\lambda}\boldsymbol{r}_{1:t}\right)$$

where $\lambda = \sum_{\tau=1}^{t-1}\gamma^{\tau-1}$ acts as a scaling factor, $\boldsymbol{r}_{1:t} = \sum_{\tau=1}^{t-1}\gamma^{\tau-1}\boldsymbol{r}_\tau$ denotes the discounted cumulative reward received so far, which is reset to zero at the beginning of an episode. Then we modify the target Q-value as follows:

$$\hat{\boldsymbol{Q}}_{\boldsymbol{\theta}}(\bar{s}_t, a) = \boldsymbol{r}_t + \gamma\hat{\boldsymbol{Q}}_{\boldsymbol{\theta}'}(\bar{s}_{t+1}, a^*),$$

where $a^* = \mathrm{argmax}_{a'\in\mathcal{A}}\,\mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}\left(\hat{\boldsymbol{Q}}_{\boldsymbol{\theta}'}(\bar{s}_{t+1}, a')\right)$. Here the immediate reward $\boldsymbol{r}_t$ is removed from the G^{3}F computation since this signal is already included in the augmented state as part of the cumulative reward. For execution in a state s, an action in $\mathrm{argmax}_{a\in\mathcal{A}}\,\mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}\left(\hat{\boldsymbol{Q}}_{\boldsymbol{\theta}}(\bar{s}, a)\right)$ is chosen. This algorithm is called G^{3}F-*CDQN*.

G^{3}F-***CSDQN***. Since stochastic policies may dominate deterministic ones (Section 4.3.3), we may improve the performance of G^{3}F-CDQN by learning a stochastic policy. We describe how to achieve this next.

First, we describe how G^{3}F-DQN can be modified to learn stochastic policies. The target Q-value is changed to:

$$\hat{\boldsymbol{Q}}_{\boldsymbol{\theta}}(s, a) = \boldsymbol{r} + \gamma\hat{\boldsymbol{Q}}^*_{\boldsymbol{\theta}'}(s', \cdot),$$

where $\hat{\boldsymbol{Q}}^*_{\boldsymbol{\theta}'}(s', \cdot) = \sum_{a'\in\mathcal{A}}\pi^*(a'|s')\hat{\boldsymbol{Q}}_{\boldsymbol{\theta}'}(s', a')$ denotes an estimated Q-value achieved at a next state by a policy π^*, which is defined as:

$$\pi^*(\cdot|s') = \underset{\pi}{\mathrm{argmax}}\,\mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}(\boldsymbol{r} + \gamma\sum_{a'\in\mathcal{A}}\pi(a'|s')\hat{\boldsymbol{Q}}_{\boldsymbol{\theta}'}(s', a')) \quad (10)$$

This reformulation assumes that in the next state, the best stochastic policy is chosen (in contrast to the deterministic greedy policy in DQN or G^{3}F-DQN). For execution in a state s, an action is sampled from $\pi^*(\cdot|s)$ in $\mathrm{argmax}_\pi\,\mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}(\sum_{a'\in\mathcal{A}}\pi(a'|s')\hat{\boldsymbol{Q}}_{\boldsymbol{\theta}}(s', a'))$.

Problem (10) is a non-linear convex optimization problem that can be solved via linear programming [35]:

$$\begin{aligned} max \sum_{k=1}^{D}\frac{k}{n}\boldsymbol{w}'_k x_k - \sum_{k=1}^{D}\sum_{i=1}^{D}\boldsymbol{w}'_k\boldsymbol{p}_i d_{ik} \quad (11)\\ s.t.\ x_k - d_{ik} \le \boldsymbol{r}_i + \gamma\boldsymbol{y}_i,\ \forall i,k = 1,\ldots,D\\ \boldsymbol{y} = \sum_{a'\in\mathcal{A}}\pi(a'|s')\hat{\boldsymbol{Q}}_{\boldsymbol{\theta}'}(s', a')\\ 0 \le \pi(a'|s') \le 1,\ \sum_{a'\in\mathcal{A}}\pi(a'|s') = 1\\ d_{ik} \ge 0,\ \forall i,k = 1,\ldots,D \end{aligned}$$

where $\boldsymbol{w}'_k = D(\boldsymbol{w}_k - \boldsymbol{w}_{k+1})$ for $k = 1,\ldots,D-1$, $\boldsymbol{w}'_D = D\boldsymbol{w}_D$, x_k's and d_{ik}'s are additional variables introduced to linearize the original non-linear optimization problem. Finally, by introducing state augmentation like in G^{3}F-CDQN, we can formulate a novel algorithm called G^{3}F-*CSDQN*, which can learn fair stochastic policies for augmented states. Note that although one may expect a better performance from this new algorithm, G^{3}F-CDQN may still be useful in domains where deterministic policies are favored (e.g., robotics).

5.2 Policy Gradient Methods

Although usually less sample-efficient than DQN-based algorithms, policy gradient methods constitute another natural choice for solving Problem (9). Following the work by Siddique et al. [43], we show how to extend two actor-critic (AC) methods: A2C [24] and PPO [40] to solve our problem. We call our new algorithms: G^{3}F-A2C and G^{3}F-PPO respectively. A nice feature of those methods is that they can directly learn a stochastic policy. Note that other policy gradient methods could be extended in a similar fashion.

G^{3}F-***A2C***. To reduce the variance of the estimation of the policy gradient (4), A2C uses a control variate method where a state-dependent baseline is subtracted from $\boldsymbol{Q}_{\pi_{\boldsymbol{\theta}}}$. Using $v(s)$ as a baseline yields the advantage function, which is estimated in A2C by $\boldsymbol{A}_{\mathrm{A2C}}(s_t, a_t) = \sum_{t=1}\gamma^{t-1}R_t - v(s_t)$ where R_t is the immediate reward obtained at time step t. In A2C, the actor update derives from the policy gradient obtained from:

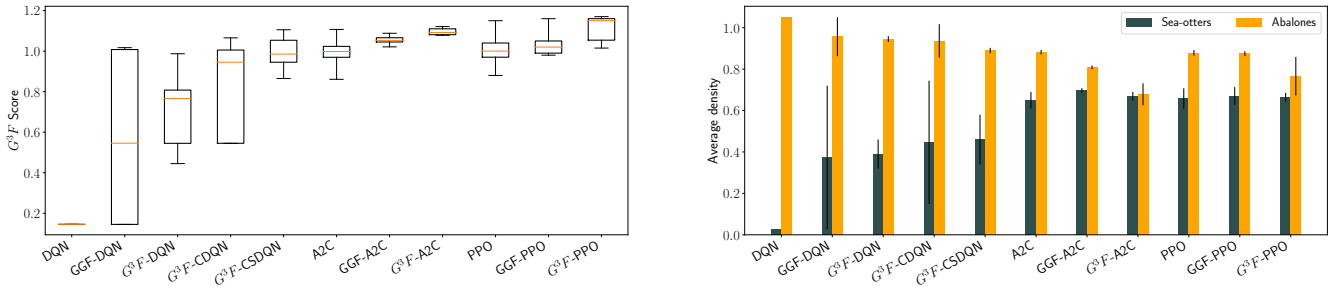
$$\boldsymbol{J}_{\mathrm{A2C}}(\boldsymbol{\theta}) = \mathbb{E}_{s\sim d_\pi, a\sim\pi_{\boldsymbol{\theta}}(\cdot|s)}\left[\boldsymbol{A}_{\mathrm{A2C}}(s, a)\right].$$

For G^{3}F-A2C, the policy gradient is formulated as follows:

$$\begin{aligned} &\nabla_{\boldsymbol{\theta}}\mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}(\boldsymbol{J}_{\mathrm{A2C}}(\boldsymbol{\theta})) && (12)\\ &= \nabla_{\boldsymbol{J}_{\mathrm{A2C}}(\boldsymbol{\theta})}\mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}(\boldsymbol{J}_{\mathrm{A2C}}(\boldsymbol{\theta}))\cdot\nabla_{\boldsymbol{\theta}}\boldsymbol{J}_{\mathrm{A2C}}(\boldsymbol{\theta}) && (13)\\ &= \boldsymbol{\omega}_\sigma^{\intercal}\cdot\nabla_{\boldsymbol{\theta}}\boldsymbol{J}_{\mathrm{A2C}}(\boldsymbol{\theta}), && (14) \end{aligned}$$

where $\nabla_{\boldsymbol{J}_{\mathrm{A2C}}(\boldsymbol{\theta})}\mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}(\boldsymbol{J}_{\mathrm{A2C}}(\boldsymbol{\theta}))\in\mathbb{R}^D$ is the gradient of function $\mathrm{G^3F}_{\boldsymbol{p},\boldsymbol{w}}$ with respect to its components and $\nabla_{\boldsymbol{\theta}}\boldsymbol{J}_{\mathrm{A2C}}(\boldsymbol{\theta})\in\mathbb{R}^{D\times N}$ (N being the number of policy parameters) represents the classic policy gradient extended to the vector case.

G^{3}F-***PPO***. Following the design of PPO [40], the advantage is estimated with λ-returns. Formally, the estimated advantage function $\boldsymbol{A}_{\mathrm{PPO}}(s, a)$ can be written as $\boldsymbol{A}_{\mathrm{PPO}}(s_t, a_t) = \sum_t(\gamma\lambda)^{t-1}\delta_t$ where $\delta_t = R_t + \gamma v(s_{t+1}) - v(s_t)$. A similar clipped surrogate objective function can be formulated to guide policy training. Denoted $\boldsymbol{J}_{\mathrm{PPO}}(\boldsymbol{\theta})$,



(a) G^{3}F **score during testing.**

(b) Population densities during testing.

Figure 2: Performances of DQN, A2C, PPO and their GGF, G^{3}F **counterparts in SC. The weight** $\boldsymbol{p}$ **is set to** $(0.9, 0.1)$ **for** G^{3}F **algorithms.**

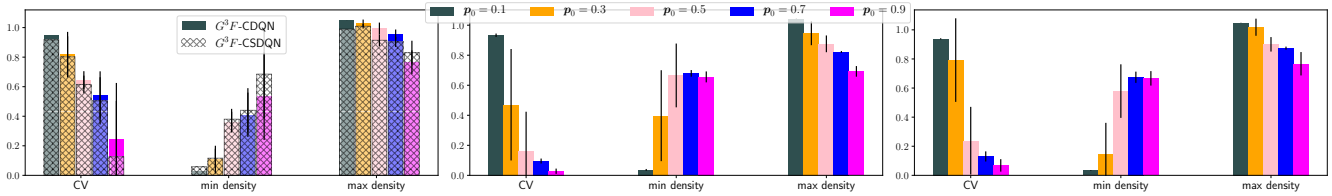


Figure 3: Effects of using different weights for $\boldsymbol{p}$ **in** G^{3}F **algorithms. CV, minimum and maximum densities of** G^{3}F**-CDQN and** G^{3}F**-CSDQN (left),** G^{3}F**-A2C (middle), and** G^{3}F**-PPO (right) during testing in SC.**

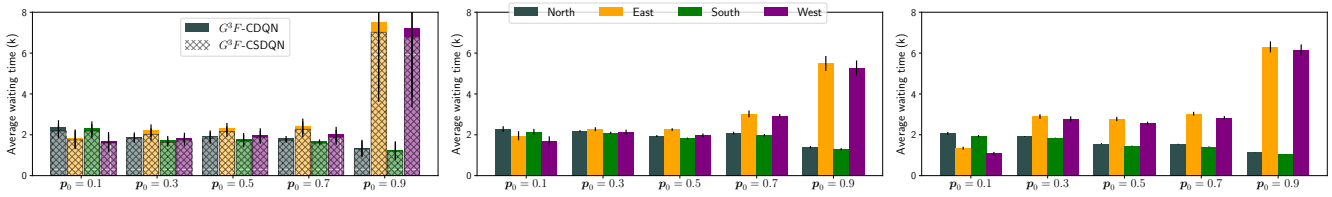


Figure 4: Effects of using different weights for $\boldsymbol{p}$ **in** G^{3}F **algorithms. Individual waiting times of** G^{3}F**-CDQN and** G^{3}F**-CSDQN (left),** G^{3}F**-A2C (middle), and** G^{3}F**-PPO (right) during testing in TL.**

it is defined so as to limit policy changes after an update:

$$\mathbb{E}_{s\sim \boldsymbol{d}_\pi, a\sim\pi_{\boldsymbol{\theta}}(\cdot|s)}\left[\min(\rho_{\boldsymbol{\theta}} \boldsymbol{A}_{\text{PPO}}(s,a), \bar{\rho}_{\boldsymbol{\theta}} \boldsymbol{A}_{\text{PPO}}(s,a))\right], \quad (15)$$

where $\rho_{\boldsymbol{\theta}} = \frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_b(a|s)}$ is an importance sampling weight, π_b is the behavior policy generating the training data, $\bar{\rho}_{\boldsymbol{\theta}} = \text{clip}(\rho_{\boldsymbol{\theta}}, 1-\epsilon, 1+\epsilon)$ is a clipped weight, and ϵ is a hyperparameter to control how much the current policy can change. The policy gradient for G^{3}F-PPO can then be obtained by replacing $\boldsymbol{J}_{\text{A2C}}$ in (12) with $\boldsymbol{J}_{\text{PPO}}$.

6 EXPERIMENTAL RESULTS

We experimentally evaluated our algorithms (with relevant baselines) in the same three domains as in Siddique et al. [43] to help with comparability. Those domains are: (i) Species conservation (SC), (ii) Traffic light control (TL), and (iii) Data center control (DC). We provide below a short description of those domains.

The first domain (SC) [8] simulates an ecological conservation problem in which two species—an endangered species (sea otters) and its prey (northern abalone)—interact with one another, potentially leading to the extinction of some species. In this domain, a state is comprised of the population level of the two species. The action space consists of five actions including *do nothing*, *introduce sea otters*, *enforce antipoaching*, *control sea otters*, and *one-half antipoaching and one-half control sea otters*. The reward vector is composed of each species' density. Fairness is expressed over the two species ($D = 2$) and can be understood as both species remaining alive and having a balanced population. Because densities may not be comparable directly, using equal weights for $\boldsymbol{p}$ may not be suitable. In that case, G^{3}F may be beneficial.

The second domain (TL) is a traffic light control problem in which an agent controls the traffic lights at a single intersection to optimize traffic flow. To simulate the traffic, we use the Simulation of Urban Mobility (SUMO)[3]. A state in this domain is composed of the waiting times and densities of cars waiting at the intersection. An

[3]https://github.com/eclipse/sumo

action amounts to selecting the next traffic-light phase. We assumed four phases: NSL, NSSR, EWL, and EWSR, with NSR representing the (north-south left) phase when the green light is assigned to the left lanes of roads approaching from the north and south, NSSR representing the (north-south straight and right) phase when the green light is assigned to the straight and right lanes of roads approaching from the north and south, and so on. Typically, the goal in this domain is to minimize the total waiting time of all cars stopped at the intersection. However, we consider fairness over each direction at the intersection (i.e., D=4). More specifically, we assume that some lanes will be given preferential treatment (e.g., due to morning rush, traffic flows are unbalanced) and that the waiting times for cars in these lanes will be optimized with higher priorities, while other lanes with equal preferences will be treated fairly.

The third and last domain (DC) is a data center traffic control problem [39], which involves connecting a large number of computers according to some network topology. In particular, we consider a network with a fat-tree topology, which connects 16 computers via 20 switches. A state is composed of each computer network information. A continuous action corresponds to the allocation of bandwidth for each host. The vector reward is calculated by penalizing the bandwidths per host by the sum of queue lengths. In this domain, fairness can be expressed with respect to the number of hosts (e.g., D = 16).

The three experimental domains are listed in ascending order of increasing number of objectives and complexity. The action spaces in the SC and TL domains are discrete, whereas the action space in the DC domain is continuous. As usual practice, we set weights $\boldsymbol{w}_i = \frac{1}{2^i}, i = 0, ..., D-1$. We will experiment with various $\boldsymbol{p}$ settings to demonstrate its effects on G^{3}F. Again, recall that weights $\boldsymbol{p}$ and $\boldsymbol{w}$ are problem-dependent and should be set by the system designer to achieve the level of fairness she desires. All experimental results are averaged over 10 runs with different seeds.

On these three domains, we have run an extensive set of experiments to answer a series of questions. We list below the main questions with their empirical answers. More experimental results can be found in the full version of this paper.

Does* G^{3}F *algorithms yield higher* G^{3}F *score than GGF or standard algorithms? This first question is a sanity check to verify that our new algorithms do optimize G^{3}F. We compare the G^{3}F scores of DQN, A2C, and PPO with their GGF and G^{3}F counterparts in the SC domain. The G^{3}F scores are obtained by applying $\text{G}^3\text{F}_{\boldsymbol{p},\boldsymbol{w}}$ on the empirical average vector returns of trajectories sampled with the learned policies during the test phase. Figure 2a depicts the distribution of this score for the policies learned by DQN, A2C, PPO, and their GGF and G^{3}F extensions. The weight $\boldsymbol{p}$ is set to $(0.9, 0.1)$ for G^{3}F methods, where the first component corresponds to sea otters. As expected, the GGF algorithms can find a fairer solution than their original versions, thus have a higher G^{3}F score. However, the G^{3}F algorithms show an even higher score than both their GGF and original counterparts, indicating that fairness with priority set by $\boldsymbol{p}$ was better achieved, as can be seen in Figure 2b. For instance, G^{3}F-A2C nearly balances the densities thanks to the higher priority given to sea otters. Recall that naturally the density of abalone would be much larger [43].

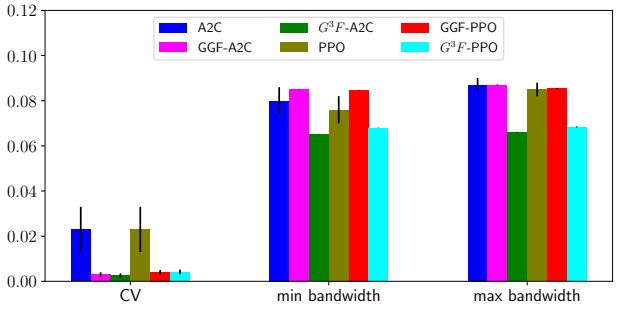
As the G^{3}F score does not directly show the vector compositions, plots of non-aggregated accumulated densities estimated during the testing phase are also presented (Figure 2b), which is simple to do for the SC domain because it is bi-objective. Compared to the standard or GGF counterparts, optimizing G^{3}F with a higher priority given to sea otters achieves more balanced individual densities. This suggests that a non-uniform $\boldsymbol{p}$ may help correct advantages conferred to some users by the environment.

What is the effect of training a policy with different weights for $\boldsymbol{p}$? To answer this question, we evaluate the performances of the G^{3}F algorithms with different weights for $\boldsymbol{p}$ in the SC and TL domains. Figure 3 shows the performance of G^{3}F-CDQN, G^{3}F-CSDQN, G^{3}F-A2C, and G^{3}F-PPO during the testing phase in terms of *Coefficient of Variation* (CV), minimum and maximum density. Recall that CV is defined as the ratio of the standard deviation to the mean. It can be interpreted as a simple measure of inequality, with lower CV values implying more balanced solutions. For experiments in the SC domain, we increase the preference weight of first objective $\boldsymbol{p}_0$ from 0.1 to 0.9 (i.e., $\boldsymbol{p}_1$ decreases from 0.9 to 0.1, correspondingly). As a result, the density of the sea otter increases, resulting in lower CV, higher minimum density, and lower maximum density for all G^{3}F algorithms.

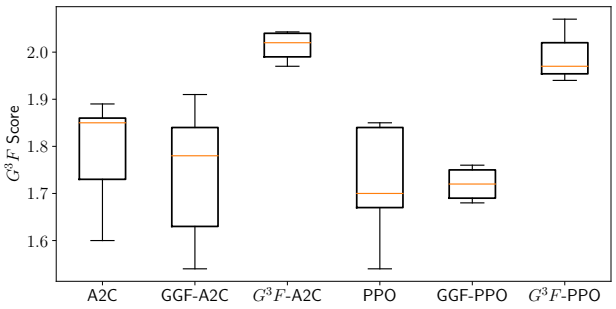
In the TL domain, we vary weight $\boldsymbol{p}_0$ (assigned to North), while the remaining weight is assigned uniformly over the remaining three components (directions) of $\boldsymbol{p}$. As shown in Figure 4, waiting times of cars coming from lanes with higher weights are shorter than those coming from lanes with lower weights. It can be observed that the waiting times of cars coming from the north and south are close, despite the fact that they are assigned different weights. This is due to the fact that the agent's action can affect two lanes at the same time in this case. For example, an action *NSL* corresponds to the phase when the left lanes of north and south are given a green light and cars can only turn left during this phase. As a result, optimizing the waiting time in one lane will have an effect on the opposite lane as well.

The above results show that by appropriately adjusting the weights $\boldsymbol{p}$, we can achieve desired control over multiple objectives.

When more weight is given to one objective and equal weights are assigned to the other objectives, does the "equal treatment of equals" principle still hold? The previous discussion in the TL domain suggests that this may not always be the case, due to the inherent structure of the control problem. It is however interesting to answer this question when less or no dependence between objectives is expected. We therefore turn to the DC domain where there are 16 objectives in total. In this domain, the first objective is given a weight of $\frac{1}{4}$, and the other objectives are assigned equal weights, i.e., $\frac{1}{20}$. Figure 5a illustrates the performances of standard deep RL algorithms and their GGF/G^{3}F counterparts in terms of CV (w.r.t the objectives with identical weights, i.e., the first objective is excluded in this statistic), minimum, and maximum bandwidths. As expected, the GGF algorithms have a lower CV than standard RL algorithms, which indicates that they can find fairer policies than their original versions. Compared to standard or GGF versions of A2C and PPO, the G^{3}F counterparts have lower minimum and maximum bandwidths since more weight is given to the first

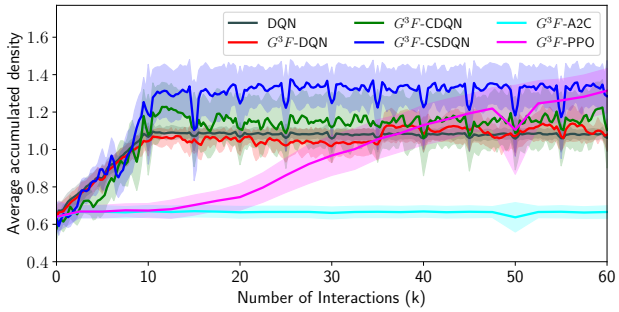


(a) CV (w.r.t the objectives with equal importance weights), minimum and maximum bandwidths.

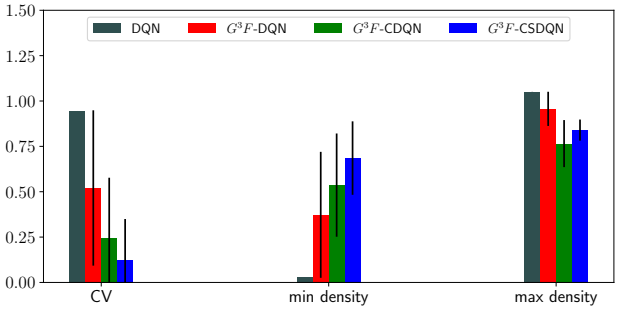


(b) G^{3}F score.

Figure 5: Performances of A2C, PPO and their GGF, G^{3}F counterparts during testing in DC.



(a) Accumulated densities during training.



(b) CV, minimum and maximum densities during testing.

Figure 6: Performances of DQN-based algorithms in SC. The weight p is set to $(0.9, 0.1)$ for G^{3}F algorithms.

objective. However, we notice that the objectives with identical weights are treated fairly, as indicated by lower CVs than standard RL algorithms, which validates that the "equal treatment of equals" principle still hold in this case.

Does considering past discounted reward or learning a stochastic policy help in DQN-based algorithms? For this question, we compare all our DQN variants in the SC and TL domains to investigate the benefits of considering past discounted reward or stochastic policies. Figures 2a, 6a and 6b show the performances of those algorithms in the SC domain. Note that while Figure 6a plots the training curves within 60k interactions, the AC methods are indeed trained with 600k interactions for convergence before testing. These figures show that moving from DQN, G^{3}F-DQN, G^{3}F-CDQN, to G^{3}F-CSDQN nearly always yields an increase in terms of average density (more efficient), a decrease in terms of CV (more equitable), an increase in terms of min density (more equitable), and an increase in terms of G^{3}F (fairer). This latter point experimentally confirms the theoretical discussion about the optimality of stochastic policies in Section 4.3.

While the above results are obtained with non-uniform $\boldsymbol{p}$, we also run experiments in the GGF setting (i.e., with uniform $\boldsymbol{p}$). The results show that our proposed G^{3}F-CDQN (i.e., GGF-CDQN) can find better solutions than GGF-DQN, suggesting that our novel DQN-based methods also apply to usual fair optimization problem without any preferential treatment. Similar conclusion can be drawn for the TL domain as well.

Interestingly, G^{3}F-CDQN and G^{3}F-CSDQN outperform DQN in terms of average density, which is exactly what is optimized by DQN. This is explained by the fact that this domain is actually partially observable. In addition, Figure 6a also includes the training curves of the AC methods (A2C and PPO) for comparison. It can be observed that the DQN-based variants learn much faster than the AC methods in terms of number of interactions. Therefore, the DQN-based variants would become more preferable choices when the sample efficiency is important.

7 CONCLUSION

We investigated the fair optimization problem with preferential treatment in RL. We presented several extensions of deep RL algorithms to tackle it, and notably proposed a novel state-augmented DQN-based method, which can be adapted to learn either deterministic or stochastic policies. Extensive experimental results on several domains were provided for validation. As future work, we plan to investigate the multi-agent extension of our new problem.

REFERENCES

[1] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. 2018. A reductions approach to fair classification. In *International Conference on Machine Learning*. PMLR, 60–69.

[2] Edoardo Amaldi, Stefano Coniglio, Luca G. Gianoli, and Can Umut Ileri. 2013. On Single-Path Network Routing Subject to Max-Min Fair Flow Allocation. *Electronic Notes in Discrete Mathematics* 41 (June 2013), 543–550.

[3] Xiaohui Bei, Shengxin Liu, Chung Keung Poon, and Hongao Wang. 2022. Candidate selections with proportional fairness constraints. *Autonomous Agents and Multi-Agent Systems* 36, 1 (2022), 1–32.

[4] Aurélie Beynier, Yann Chevaleyre, Laurent Gourvès, Ararat Harutyunyan, Julien Lesca, Nicolas Maudet, and Anaëlle Wilczynski. 2019. Local envy-freeness in house allocation problems. *Autonomous Agents and Multi-Agent Systems* 33, 5 (2019), 591–627.

[5] Steven J. Brams and Alan D. Taylor. 1996. *Fair Division: From Cake-Cutting to Dispute Resolution.* Cambridge University Press.

[6] Steven J. Brams and Alan D. Taylor. 1996. *Fair Division: From Cake-Cutting to Dispute Resolution.* Cambridge University Press.

[7] Róbert Busa-Fekete, Balázs Szörényi, Paul Weng, and Shie Mannor. 2017. Multi-objective bandits: Optimizing the generalized gini index. In *International Conference on Machine Learning*. PMLR, 625–634.

[8] Iadine Chadès, Janelle MR Curtis, and Tara G Martin. 2012. Setting realistic recovery targets for two interacting endangered species, sea otter and northern abalone. *Conservation Biology* 26, 6 (2012), 1016–1025.

[9] Mithun Chakraborty, Ayumi Igarashi, Warut Suksompong, and Yair Zick. 2021. Weighted envy-freeness in indivisible item allocation. *ACM Transactions on Economics and Computation (TEAC)* 9, 3 (2021), 1–39.

[10] Jingdi Chen, Yimeng Wang, and Tian Lan. 2021. Bringing Fairness to Actor-Critic Reinforcement Learning for Network Utility Optimization. In *INFOCOM*.

[11] Yann Chevaleyre, Paul E Dunne, Michel Lemaître, Nicolas Maudet, Julian Padget, Steve Phelps, and Juan A Rodríguez-aguilar. 2006. Issues in Multiagent Resource Allocation. *Computer* 30 (2006), 3–31.

[12] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. 2017. Fair Clustering Through Fairlets. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc.

[13] Yinlam Chow and Mohammad Ghavamzadeh. 2014. Algorithms for CVaR optimization in MDPs.

[14] Cyrus Cousins. 2021. An axiomatic theory of provably-fair welfare-centric machine learning. *Advances in Neural Information Processing Systems* 34 (2021), 16610–16621.

[15] Virginie Do and Nicolas Usunier. 2022. Optimizing generalized Gini indices for fairness in rankings.

[16] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through Awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. 214–226.

[17] Ganesh Ghalme, Vineet Nair, Vishakha Patil, and Yilun Zhou. 2022. Long-Term Resource Allocation Fairness in Average Markov Decision Process (AMDP) Environment. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 525–533.

[18] Hoda Heidari, Claudio Ferrari, Krishna Gummadi, and Andreas Krause. 2018. Fairness behind a veil of ignorance: A welfare analysis for automated decision making. *Advances in Neural Information Processing Systems* 31 (2018).

[19] Shahin Jabbari, Matthew Joseph, Michael Kearns, Jamie Morgenstern, and Aaron Roth. 2017. Fairness in reinforcement learning. In *International conference on machine learning*. PMLR, 1617–1626.

[20] Jiechuan Jiang and Zongqing Lu. 2019. Learning fairness in multi-agent systems. *Advances in Neural Information Processing Systems* 32 (2019).

[21] Y. Liu and S. Koenig. 2005. Risk-Sensitive Planning with One-Switch Utility Functions: Value Iteration. In *AAAI*. AAAI, 993–999.

[22] László Lovász. 1983. Submodular functions and convexity. In *Mathematical programming the state of the art*. Springer, 235–257.

[23] Debmalya Mandal and Jiarui Gan. 2022. Socially Fair Reinforcement Learning. *arXiv preprint arXiv:2208.12584* (2022).

[24] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *ICML*.

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (2015), 529–533.

[26] Hervé Moulin. 2004. *Fair division and collective welfare.* MIT press.

[27] H. Moulin. 2004. *Fair Division and Collective Welfare.* MIT Press.

[28] Razieh Nabi, Daniel Malinsky, and Ilya Shpitser. 2019. Learning Optimal Fair Policies. In *ICML*.

[29] Lorenzo Nardi and Cyrill Stachniss. 2019. Uncertainty-aware path planning for navigation on road networks using augmented MDPs. In *ICRA*.

[30] Somjit Nath, Mayank Baranwal, and Harshad Khadilkar. 2021. Revisiting State Augmentation methods for Reinforcement Learning with Stochastic Delays. In *CIKM*.

[31] Arnie Neidhardt, Hanan Luss, and K. R. Krishnan. 2008. Data Fusion and Optimal Placement of Fixed and Mobile Sensors. In *2008 IEEE Sensors Applications Symposium*.

[32] Viet Hung Nguyen and Paul Weng. 2017. An Efficient Primal-Dual Algorithm for Fair Combinatorial Optimization Problems. In *COCOA*.

[33] Włodzimierz Ogryczak, Hanan Luss, Michał Pióro, Dritan Nace, and Artur Tomaszewski. 2014. Fair optimization and networks: A survey. *Journal of Applied Mathematics* 2014 (2014).

[34] Wlodzimierz Ogryczak, Patrice Perny, and Paul Weng. 2013. A compromise programming approach to multiobjective Markov decision processes. *International Journal of Information Technology & Decision Making* 12, 05 (2013), 1021–1053.

[35] Włodzimierz Ogryczak and Tomasz Śliwiński. 2007. On optimization of the importance weighted OWA aggregation of multiple criteria. In *International Conference on Computational Science and Its Applications*. Springer, 804–817.

[36] Włodzimierz Ogryczak and Tomasz Śliwiński. 2010. On solving optimization problems with ordered average criteria and constraints. *Fuzzy Optimization: Recent Advances and Applications* (2010), 209–230.

[37] M.L. Puterman. 1994. *Markov decision processes: discrete stochastic dynamic programming*. Wiley.

[38] John Rawls. 1971. *The Theory of Justice.* Havard university press.

[39] Fabian Ruffy, Michael Przystupa, and Ivan Beschastnikh. 2019. Iroko: A Framework to Prototype Reinforcement Learning for Data Center Traffic Control. In *Workshop on ML for Systems at NeurIPS*. http://arxiv.org/abs/1812.09975

[40] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). arXiv:1707.06347 http://arxiv.org/abs/1707.06347

[41] Saeed Sharifi-Malvajerdi, Michael Kearns, and Aaron Roth. 2019. Average Individual Fairness: Algorithms, Generalization and Experiments. In *Advances in Neural Information Processing Systems*.

[42] Huaizhou Shi, R. Venkatesha Prasad, Ertan Onur, and I. G. M. M. Niemegeers. 2014. Fairness in Wireless Networks:Issues, Measures and Challenges. *IEEE Communications Surveys & Tutorials* 16, 1 (2014), 5–24.

[43] Umer Siddique, Paul Weng, and Matthieu Zimmer. 2020. Learning Fair Policies in Multi-Objective (Deep) Reinforcement Learning with Average and Discounted Rewards. In *ICML*.

[44] Ashudeep Singh and Thorsten Joachims. 2019. Policy Learning for Fairness in Ranking. In *Advances in Neural Information Processing Systems*.

[45] Aivar Sootla, Alexander I Cowen-Rivers, Taher Jafferjee, Ziyan Wang, David H Mguni, Jun Wang, and Haitham Ammar. 2022. Sauté RL: Almost surely safe reinforcement learning using state augmentation. In *ICML*.

[46] Till Speicher, Hoda Heidari, Nina Grgic-Hlaca, Krishna P Gummadi, Adish Singla, Adrian Weller, and Muhammad Bilal Zafar. 2018. A unified approach to quantifying algorithmic unfairness: Measuring individual &group unfairness via inequality indices. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2239–2248.

[47] Ankang Sun, Bo Chen, and Xuan Vinh Doan. 2021. Connections between fairness criteria and efficiency for allocating indivisible chores. *arXiv preprint arXiv:2101.07435* (2021).

[48] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 2000. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *NIPS*.

[49] Min Wen, Osbert Bastani, and Ufuk Topcu. 2021. Algorithms for Fairness in Sequential Decision Making. In *ICML*.

[50] Paul Weng. 2019. Fairness in reinforcement learning. *arXiv preprint arXiv:1907.10323* (2019).

[51] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, Krishna P. Gummadi, and Adrian Weller. 2017. From Parity to Preference-Based Notions of Fairness in Classification. In *Advances in Neural Information Processing Systems*.

[52] Xueru Zhang and Mingyan Liu. 2021. Fairness in learning-based sequential decision algorithms: A survey. In *Handbook of Reinforcement Learning and Control*. Springer, 525–555.

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