

731 A DICE-RL framework

732 In this section, we introduce the DICE-RL framework along with its corresponding offline single-
 733 objective reinforcement learning algorithm, OptiDICE, as proposed in [22]. Our algorithm can
 734 be viewed as a multi-objective extension of OptiDICE. The DICE-RL framework is an offline RL
 735 framework where return maximization is regularized with an f -divergence between the stationary
 736 distribution of the learned policy d and the empirical distribution d_D , solving:

$$\begin{aligned} & \max_{d \geq 0} \sum_{s,a} d(s,a) r(s,a) - \beta \sum_{s,a} d_D(s,a) f\left(\frac{d(s,a)}{d_D(s,a)}\right) \\ & \text{s.t. } \sum_a d(s,a) = (1-\gamma)p_0(s) + \gamma \sum_{\bar{s}, \bar{a}} T(s|\bar{s}, \bar{a}) d(\bar{s}, \bar{a}), \quad \forall s \end{aligned}$$

737 where the Bellman flow constraints ensure that the optimal $d^*(s,a)$ constitutes a valid stationary
 738 distribution. Lagrangian dual of the convex optimization problem is given by,

$$\max_{d \geq 0} \min_{\nu} \sum_{s,a} d(s,a) r(s,a) + \sum_s \nu(s) \mathcal{F}_d(s) - \beta \sum_{s,a} d_D(s,a) f\left(\frac{d(s,a)}{d_D(s,a)}\right)$$

739 where $\mathcal{F}_d(s) = (1-\gamma)p_0(s) + \gamma \sum_{\bar{s}, \bar{a}} T(s|\bar{s}, \bar{a}) d(\bar{s}, \bar{a}) - \sum_a d(s,a)$. We reparameterize the
 740 stationary distribution as $d(s,a) = w(s,a) d_D(s,a)$ and express the dual in terms of the importance
 741 weights w , using the identity $\sum_s T(s|\bar{s}, \bar{a}) \nu(s) \sum_{\bar{s}, \bar{a}} d(\bar{s}, \bar{a}) = \sum_{s,a} d(s,a) \sum_{s'} T(s'|s,a) \nu(s')$.
 742 This yields the following optimization problem:

$$\max_{w \geq 0} \min_{\nu} L(w, \nu) := \mathbb{E}_{s \sim p_0} [(1-\gamma)\nu(s)] + \mathbb{E}_{(s,a) \sim d_D} [w(s,a) e_{\nu}(s,a) - \beta f(w(s,a))]$$

743 where $e_{\nu}(s,a) = r(s,a) + \gamma \sum_{s'} T(s'|s,a) \nu(s') - \nu(s) \forall s, a$.

744 Strong duality holds by Slater's condition, since the problem is convex and a valid stationary
 745 distribution exists in the strictly feasible set. Therefore, the order of optimization can be swapped to
 746 $\min_{\nu} \max_{w \geq 0}$. Optimal $w^*(s,a)$ is computed from the first-order condition given by:

$$\frac{\partial L(w, \nu)}{\partial w(s,a)} = \mathbb{E}_{(s,a) \sim d_D} [e_{\nu}(s,a) - \beta f'(w(s,a))] = 0$$

747 where $w_{\nu}^*(s,a) = \max\left(0, (f')^{-1}\left(\frac{e_{\nu}(s,a)}{\beta}\right)\right)$. By plugging $w^*(s,a)$ in $L(w, \nu)$ results in the
 748 following ν loss of OptiDICE.

$$\min_{\nu} \mathbb{E}_{s \sim p_0} [(1-\gamma)\nu(s)] + \mathbb{E}_{(s,a) \sim d_D} \left[\beta f_0^* \left(\frac{e_{\nu}(s,a)}{\beta} \right) \right]$$

749 where $f_0^*(y) := \max_{x \geq 0} xy - f(x)$.

750 After minimizing over ν , the optimal stationary distribution is given by $d^*(s,a) = w_{\nu^*}^*(s,a) d_D(s,a)$.
 751 This distribution can then be used to recover the optimal policy that induces $d^*(s,a)$ via $\pi^*(a|s) =$
 752 $\frac{d^*(s,a)}{\sum_a d^*(s,a)}$ for all s, a , or through a policy extraction method such as weighted behavior cloning.

753 B Counterexample

754 In this section, we present a counterexample demonstrating that applying the optimal implicit weights
 755 μ^* from Fair MORL (P2) to Linear MORL (P3) does not recover the optimal policy of Fair MORL
 756 (P2). Consider an MDP with a single state s and a terminal state reached immediately after taking an
 757 action. There are two available actions, $a \in \{a_1, a_2\}$, with corresponding stationary distributions
 758 $d(s, a_1)$ and $d(s, a_2)$. Each action yields a reward vector consisting of rewards for objectives A and
 759 B : $\mathbf{r}(s, a_1) = [r_A(s, a_1), r_B(s, a_1)] = [1, 4]$ and $\mathbf{r}(s, a_2) = [r_A(s, a_2), r_B(s, a_2)] = [3, 1]$.

760 Using this setup, we construct the corresponding (P1) optimization problem, while assuming Nash
 761 social welfare ($u_i = \log(x) \forall i$):

$$\begin{aligned} \text{(P1): } & \max_{d \geq 0} \log(d(s, a_1) + 3d(s, a_2)) + \log(4d(s, a_1) + d(s, a_2)) \\ & \text{s.t. } d(s, a_1) + d(s, a_2) = 1 \end{aligned}$$

where $\log(x)$ is applied to the returns of each objective. An equivalent optimization (P2) is given by:

$$\begin{aligned} \text{(P2): } \max_{d \geq 0, k} \quad & \log(k_1) + \log(k_2) \\ \text{s.t. } \quad & d(s, a_1) + d(s, a_2) = 1 \\ & d(s, a_1) + 3d(s, a_2) = k_1 \\ & 4d(s, a_1) + d(s, a_2) = k_2 \end{aligned}$$

Its Lagrangian dual is given as,

$$\begin{aligned} \max_{d \geq 0, k} \min_{\mu, \nu} \quad & \log(k_1) + \log(k_2) + \nu(d(s, a_1) + d(s, a_2) - 1) \\ & + \mu_1(d(s, a_1) + 3d(s, a_2) - k_1) + \mu_2(4d(s, a_1) + d(s, a_2) - k_2) \end{aligned}$$

By solving the first-order conditions, the optimal Lagrange multipliers are $\mu_1^* \approx 0.5455$ and $\mu_2^* \approx 0.3636$, resulting in the optimal stationary distribution $[d^*(s, a_1), d^*(s, a_2)] = [0.5834, 0.4166]$. The optimal policy of Fair MORL (P2) is a stochastic policy that prefers action a_1 while still assigning probability to a_2 , resulting in objective returns $k^* = [1.8332, 2.7500]$. While μ is applied to the objective returns in a manner analogous to Linear MORL, we demonstrate that the two formulations are fundamentally distinct by applying the implicit preference weight μ^* to (P3):

$$\begin{aligned} \text{(P3): } \max_{d \geq 0} \quad & (1 \cdot \mu_1^* + 4 \cdot \mu_2^*)d(s, a_1) + (3 \cdot \mu_1^* + 1 \cdot \mu_2^*)d(s, a_2) \\ \text{s.t. } \quad & d(s, a_1) + d(s, a_2) = 1 \end{aligned}$$

In this case, $1 \cdot \mu_1^* + 4 \cdot \mu_2^* = 3 \cdot \mu_1^* + 1 \cdot \mu_2^* = 2.0$, indicating that all policies are the all optimal policies of (P3). As a result, any policy is optimal under Linear MORL with fixed weights μ . This demonstrates that (P2) and (P3) are distinct optimization problems.

C Proof of Proposition 1

In this section, we provide a proof of Proposition 1, showing that regularized Linear MORL (P3-reg), when using preference weights equal to the optimal dual variable μ^* from regularized Fair MORL (P2-reg), converges to the same solution as (P2-reg). To facilitate the explanation, we begin by rewriting the (P3-reg) formulation:

$$\text{(P3-reg): } \max_{d \geq 0} \sum_{s,a} d(s, a) \sum_i \mu_i^* r_i(s, a) - \beta \sum_{s,a} d_D(s, a) f\left(\frac{d(s, a)}{d_D(s, a)}\right) \quad \text{s.t. (2)}$$

The Lagrangian duals of (P2-reg) and (P3-reg) are given by:

$$\begin{aligned} \max_{d \geq 0, k} \min_{\nu, \mu} L_{\text{P2-reg}}(\nu, \mu, d, k) &:= \sum_i \mu_i \left(\sum_{s,a} d(s, a) r_i(s, a) - k_i \right) - \beta \sum_{s,a} d_D(s, a) f\left(\frac{d(s, a)}{d_D(s, a)}\right) \\ &+ \sum_i u_i(k_i) + \sum_s \nu(s) \mathcal{F}_d(s) \\ \max_{d \geq 0} \min_{\nu} L_{\text{P3-reg}}(\nu, d) &:= \sum_{s,a} d(s, a) \sum_i \mu_i^* r_i(s, a) - \beta \sum_{s,a} d_D(s, a) f\left(\frac{d(s, a)}{d_D(s, a)}\right) \\ &+ \sum_s \nu(s) \mathcal{F}_d(s) \end{aligned}$$

Assuming the optimal μ^* from (P2-reg) is given, we compute the gradient of each Lagrangian with respect to $\nu(s)$ and $d(s, a)$, and show that both share the same gradient at their optimal solutions.

$$\begin{aligned} \frac{\partial L_{\text{P2-reg}}}{\partial \nu(s)} &= \frac{\partial L_{\text{P3-reg}}}{\partial \nu(s)} = \mathcal{F}_{d^*}(s) \\ \frac{\partial L_{\text{P2-reg}}}{\partial d(s, a)} &= \frac{\partial L_{\text{P3-reg}}}{\partial d(s, a)} = \sum_i \mu_i^* r_i(s, a) - \beta \sum_{s,a} f'\left(\frac{d^*(s, a)}{d_D(s, a)}\right) + \sum_{s'} \gamma T(s'|s, a) \nu^*(s') - \nu^*(s) \end{aligned}$$

While (P3) is a linear program, (P3-reg) becomes a convex optimization problem due to regularization with a strictly convex function f , making the KKT conditions applicable. From the stationarity conditions, (P3-reg) with μ^* converges to the same optimal solution as (P2-reg).

784 C.1 Empirical evidence

785 We adapt the counterexample from Appendix B to demonstrate that, under offline regularization,
 786 Linear MORL and Fair MORL can share the same optimal solution. In the offline setting, we
 787 additionally assume a fixed data distribution over actions given by $[d_D(s, a_1), d_D(s, a_2)] = [0.7, 0.3]$.
 788 The (P2-reg) formulation of the counterexample is given by:

$$\begin{aligned} \text{(P2-reg): } \max_{d \geq 0, k} \quad & \log(k_1) + \log(k_2) - \sum_a d_D(s, a) f\left(\frac{d(s, a)}{d_D(s, a)}\right) \\ \text{s.t. } \quad & d(s, a_1) + d(s, a_2) = 1 \\ & d(s, a_1) + 3d(s, a_2) = k_1 \\ & 4d(s, a_1) + d(s, a_2) = k_2 \end{aligned}$$

789 where we adopt χ^2 -divergence $f(x) = \frac{1}{2}(x - 1)^2$. The optimal Lagrange multipliers are $\mu_1^* \approx$
 790 0.5959 and $\mu_2^* \approx 0.3352$, resulting in the optimal stationary distribution $[d^*(s, a_1), d^*(s, a_2)] =$
 791 $[0.6609, 0.3390]$. This indicates that as a_1 is more common in the data distribution than a_2 , offline RL
 792 policy of (P2-reg) favors a_1 compared to the unregularized case. This aligns with the goal of offline
 793 reinforcement learning where the optimized stationary distribution should not deviate excessively
 794 from the dataset distribution. We apply μ^* to (P3-reg).

$$\begin{aligned} \text{(P3-reg): } \max_{d \geq 0} \quad & (1 \cdot \mu_1^* + 4 \cdot \mu_2^*)d(s, a_1) + (3 \cdot \mu_1^* + 1 \cdot \mu_2^*)d(s, a_2) - \sum_a d_D(s, a) f\left(\frac{d(s, a)}{d_D(s, a)}\right) \\ \text{s.t. } \quad & d(s, a_1) + d(s, a_2) = 1 \end{aligned}$$

795 where $1 \cdot \mu_1^* + 4 \cdot \mu_2^* = 1.936$ and $3 \cdot \mu_1^* + 1 \cdot \mu_2^* = 2.123$.

796 As (P3-reg) is a convex optimization problem with strictly concave objective, the uniqueness of the
 797 optimal solution is guaranteed. The optimal stationary distribution of (P3-reg) is equivalent to that
 798 of (P2-reg) as $[d^*(s, a_1), d^*(s, a_2)] = [0.6609, 0.3390]$. This establishes the connection between
 799 regularized Linear MORL and Fair MORL theoretically and empirically.

800 D Finite Domain Experiment Setting

801 In this section, we provide a detailed explanation of the experimental setup used in Section 6. We
 802 begin by describing how the classic Four-Room environment [22, 29] and Random MDP [22, 30] are
 803 adapted for the offline multi-objective reinforcement learning (MORL) setting. We present additional
 804 visualizations of the MO-Four-Room results to further support the findings in Section 6.

805 D.1 Environment detail

806 **MO-Four-Room** In MO-Four-Room domain, three distinct goals, each associated with a separate
 807 objective. The agent starts from the initial state (orange) and navigates toward one of the goal states
 808 (green). Upon reaching a goal, it receives a one-hot reward vector: $[1, 0, 0]$ for the lower-left room,
 809 $[0, 1, 0]$ for the upper-right room, and $[0, 0, 1]$ for the lower-right room. The agent selects one of
 810 four actions, {left, right, up, down}; however, the environment is stochastic, and with a probability
 811 of 0.1, the agent transitions in a different direction than the one intended. To simulate offline RL, a
 812 dataset of 300 trajectories are collected from a uniformly random behavior policy. The experiments
 813 are conducted with $\alpha \in \{0.0, 1.0\}$, where $\alpha = 0.0$ corresponds to a policy that maximizes Utilitarian
 814 welfare, and $\alpha = 1.0$ corresponds to one that maximizes Nash social welfare. The regularization
 815 coefficient is fixed at $\beta = 0.01$ and $f(x) = 0.5(x - 1)^2$.

816 **Random MOMDP** In the Random MOMDP domain, a multi-objective Markov decision pro-
 817 cess is generated with $|\mathcal{S}| = 50$, $|\mathcal{A}| = 4$, and discount factor $\gamma = 0.95$. For each state-action
 818 pair, the next-state transitions are defined over four possible next states, with transition proba-
 819 bilities sampled from a Dirichlet distribution, $\text{Dir}(1, 1, 1, 1)$. Among the 49 states excluding the
 820 fixed initial state, three are randomly selected as goal states, and each is assigned a distinct one-
 821 hot reward vector in the same manner as in the MO-Four-Room environment. To simulate the
 822 offline RL setting, a dataset of 100 trajectories is collected using a behavior policy with an opti-
 823 mality level of 0.5. Here, optimality is defined as the normalized performance relative to a uni-
 824 formly random policy π_{unif} (optimality = 0.0) and an optimal policy π^* (optimality = 1.0). This

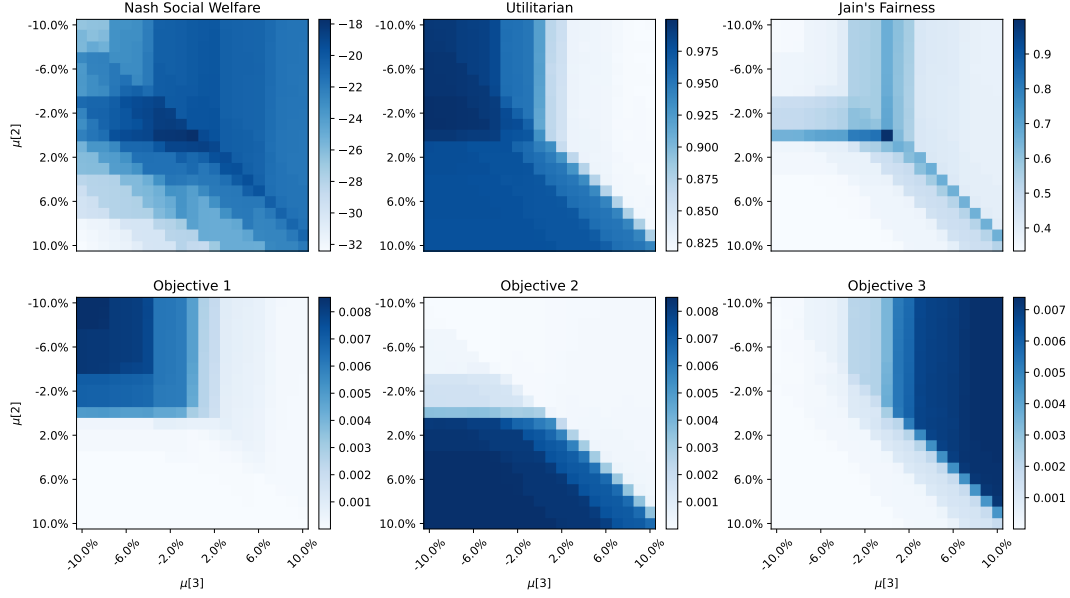


Figure 7: Visualization of FairDICE-fixed performance with different μ in MO-Four-Room. The center point is the optimal μ^* obtained by FairDICE. The x-axis and y-axis represent the degrees of perturbation applied to μ_2 and μ_3 from their optimal values.

implies that the behavior policy achieves performance halfway between that of the optimal and random policies. The experiments are repeated over 1000 seeds, with $\alpha \in \{0.0, 0.5, 1.0, 1.25\}$ and $\beta \in \{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0, 5.0, 10.0, 50.0, 100.0\}$, to provide a comprehensive analysis of the trade-off between welfare maximization and distributional shift.

D.2 Additional visulaization: MO-Four-Room

Figure 1 illustrates that FairDICE maximizes the welfare objective, resulting in behavior that is distinct from conventional MORL approaches. In this subsection, we extend the experiment from Section 6.2 to further visualize two key insights proposed in our paper: (1) μ corresponds to the preference weights used in regularized Linear MORL, and (2) these weights are implicitly optimized to maximize the welfare objective—any deviation from the optimal μ leads to a reduction in overall welfare. In Section 6.2, all preference weights were perturbed within Random MOMDP setting. While this effectively demonstrated that FairDICE selects the welfare-maximizing weights, it is not easy to visualize the consequence of deviation.

Given the optimal preference weight of FairDICE within MO-Four-Room, $\mu^* = [\mu_1, \mu_2, \mu_3]$, we perturb μ_2 and μ_3 while keeping μ_1 fixed. The perturbed weights are applied to FairDICE-fixed to obtain the corresponding optimal policy, whose performance is then evaluated as shown above.

The center point corresponds to the optimal solution of FairDICE, achieving the highest Nash social welfare. Increasing μ_2 boosts return on objective 2, while increasing μ_3 improves return on objective 3. Return on objective 1 increases when both μ_2 and μ_3 decrease. The point that maximizes Utilitarian welfare shifts toward prioritizing objective 1, but this comes at the cost of reduced Nash social welfare and lower Jain’s fairness index. These results highlight that FairDICE implicitly optimizes preference weights to maximize welfare, enabling fair behavior in the offline MORL setting.

E D4MORL Benchmark

To evaluate the efficacy of offline multi-objective reinforcement learning (MORL) algorithms, we utilize the D4MORL benchmark (with MIT License) introduced by [31]. D4MORL is the large-scale benchmark designed for offline MORL and includes high-dimensional continuous control environments derived from MuJoCo. We adopted settings and configurations of D4MORL without additional modifications.

E.1 Environments and Objectives

D4MORL comprises six environments: MO-Ant, MO-HalfCheetah, MO-Hopper, MO-Swimmer, and MO-Walker2d, each with two conflicting objectives (e.g., speed vs. energy efficiency), and MO-Hopper-3obj, which includes three objectives—making it a more challenging benchmark. Each environment is defined by multiple, often conflicting, objectives such as forward velocity, jumping stability, or energy consumption. These objectives induce trade-offs, thereby enabling the study of Pareto-optimal policy learning in continuous control settings.

We summarize the objectives for each environment in Table 2, including their physical interpretations and reward formulations. The rewards are computed based on physical quantities such as displacement, height, and control cost. Most environments include a survival bonus term r_s and penalize excessive actions via an action cost term $r_a = \sum_k a_k^2$. The time delta Δt determines the resolution of velocity and height-based rewards and is environment-specific.

Reward Terms and Environment-Specific Constants. The reward functions in Table 2 include shared terms whose values vary across environments:

- **Action penalty** (r_a): Typically defined as the squared sum of action magnitudes, i.e., $r_a = \sum_k a_k^2$. However, different environments apply distinct scaling factors: $r_a = 0.5 \sum_k a_k^2$ in MO-Ant; $r_a = 2 \times 10^{-4} \sum_k a_k^2$ in MO-Hopper.
- **Survival bonus** (r_s): A constant reward to encourage survival, set to $r_s = 1.0$ in all environments except MO-Swimmer, where it is omitted.
- **Time delta** (Δt): Represents the duration between timesteps used in computing velocities or other dynamic terms. Its value is $\Delta t = 0.05$ in MO-Ant, MO-HalfCheetah, and MO-Swimmer; $\Delta t = 0.01$ in MO-Hopper and MO-Hopper-3obj; and $\Delta t = 0.008$ in MO-Walker2d.
- **Initial height** (h_{init}): In MO-Hopper and MO-Hopper-3obj, vertical jump rewards are defined relative to a fixed starting height of $h_{\text{init}} = 1.25$.

These constants follow the environment configurations detailed in Appendix A of [31].

Table 2: Objectives in D4MORL Environments

Environment	Objective Name	Reward Description
MO-Ant	r_{vx} : velocity in x direction	$r_{vx} = \frac{x_t - x_{t-1}}{\Delta t} + r_s - r_a$
	r_{vy} : velocity in y direction	$r_{vy} = \frac{y_t - y_{t-1}}{\Delta t} + r_s - r_a$
MO-HalfCheetah	r_v : forward speed	$r_v = \min(4.0, \frac{x_t - x_{t-1}}{\Delta t}) + r_s$
	r_e : energy efficiency	$r_e = 4.0 - r_a + r_s$
MO-Hopper	r_r : forward running	$r_r = 1.5 \cdot \frac{x_t - x_{t-1}}{\Delta t} + r_s - r_a$
	r_j : vertical jumping	$r_j = 12 \cdot \frac{h_t - h_{\text{init}}}{\Delta t} + r_s - r_a$
MO-Hopper-3obj	r_r : forward running	$r_r = 1.5 \cdot \frac{x_t - x_{t-1}}{\Delta t} + r_s$
	r_j : vertical jumping	$r_j = 12 \cdot \frac{h_t - h_{\text{init}}}{\Delta t} + r_s$
	r_e : energy efficiency	$r_e = 4.0 - r_a + r_s$
MO-Swimmer	r_v : forward speed	$r_v = \frac{x_t - x_{t-1}}{\Delta t}$
	r_e : energy efficiency	$r_e = 0.3 - 0.15 \cdot r_a$
MO-Walker2d	r_v : forward speed	$r_v = \frac{x_t - x_{t-1}}{\Delta t} + r_s$
	r_e : energy efficiency	$r_e = 4.0 - r_a + r_s$

E.2 Behavioral Policy Quality

Each dataset in D4MORL is collected using:

- **Expert policy**: Selected from a large PGMORL[33]-trained ensemble to match the target preference closely.

883 • **Amateur policy:** Perturbed version of expert (35% chance of random scaling or uniform
884 sampling of actions).

885 The target preferences used during data collection are uniformly sampled from the full $(n - 1)$ -
886 dimensional simplex. This promotes diverse trade-offs across objectives and ensures that the dataset
887 covers a wide range of possible preferences. All preference vectors $\omega \in \mathbb{R}^n$ are normalized to satisfy
888 $\omega_i \geq 0$ and $\sum_i \omega_i = 1$.

889 E.3 Reward and State Normalization

890 We introduce how rewards and states are normalized in the D4MORL benchmark [31].

891 **Reward normalization.** The reward values are normalized for each objective to the $[0, 1]$ range
892 using min-max normalization:

$$r^{\text{norm}} = \frac{r - r_{\min}}{r_{\max} - r_{\min}},$$

893 where r_{\min}, r_{\max} are computed empirically from the offline dataset for each objective dimension.

894 **State normalization.** All state vectors are standardized using environment-specific statistics provided
895 by the D4MORL benchmark. Specifically, the raw state s is normalized as:

$$s^{\text{norm}} = \frac{s - \mu}{\sigma},$$

896 where μ and σ denote the per-dimension mean and standard deviation of the state distribution,
897 computed from the offline dataset.

898 F Implementation Details

899 We use the Soft- χ^2 divergence as the regularization function f in FairDICE. This function is defined
900 piecewise as:

$$f(x) = \begin{cases} x \log x - x + 1 & \text{if } x < 1 \\ \frac{1}{2}(x - 1)^2 & \text{otherwise} \end{cases}$$

901 This divergence combines the smooth behavior of KL divergence near $x = 0$ with the quadratic
902 growth of the standard χ^2 divergence for larger x .

903 The FairDICE algorithm, summarized in Algorithm 1, alternates between optimizing the dual
904 variables (ν, μ) and updating the policy π via weighted behavior cloning. Both the policy π and critic
905 ν networks are implemented as multilayer perceptrons, parameterized by ψ and θ , respectively. The
906 scalar parameters μ are updated to maximize the desired social welfare function, and we fix $\alpha = 1$ to
907 correspond to the Nash social welfare objective. The initial state distribution p_0 is estimated from the
908 offline dataset.

909 Table 3 provides a summary of our default hyperparameters. The policy and value networks are
910 constructed with three hidden layers, each containing 768 units. Optimization is performed using the
911 Adam optimizer with a learning rate of 3×10^{-4} and a discount factor of $\gamma = 0.99$. To study the effect
912 of the regularization coefficient β —which governs the trade-off between distributional robustness and
913 optimization stability—we conduct a hyperparameter sweep over $\beta \in \{1.0, 0.1, 0.01, 0.001, 0.0001\}$.

914 G Experiments Compute Resources

915 All experiments were conducted on a single machine equipped with an Intel® Xeon® Gold 6330
916 CPU (256GB RAM) and an NVIDIA RTX 3090 GPU. Training a single FairDICE policy on each
917 D4MORL task required approximately 10 to 20 minutes on average. During training, GPU memory
918 usage remained below 20GB.

919 H Impact of f-divergence on FairDICE

920 In this section, we investigate how varying β , which controls the strength of regularization, af-
921 fects FairDICE’s performance. Figure 8 reports Nash social welfare (NSW) performance for

Algorithm 1 FairDICE

Input: Offline dataset D , initial state distribution p_0 , policy π_θ , dual parameters ν_ψ, μ_i , divergence regularization parameter β , concave scalarization function u_i .

Output: Welfare-maximizing policy π_θ^*

- 1: Initialize all parameters
- 2: **while** not converged **do**
- 3: Update ν_ψ, μ to minimize:

$$\mathcal{L}_{\nu_\psi, \mu} = \mathbb{E}_{s \sim p_0}[(1 - \gamma)\nu_\psi(s)] + \mathbb{E}_{(s,a) \sim D} \left[\beta f_0^* \left(\frac{e_{\nu_\psi, \mu}(s, a)}{\beta} \right) \right] + \sum_i u_i^*(-\mu_i)$$

- 4: Compute optimal weights:

$$w_{\nu_\psi, \mu}^*(s, a) = \max \left(0, (f')^{-1} \left(\frac{e_{\nu_\psi, \mu}(s, a)}{\beta} \right) \right)$$

- 5: Update policy π_θ via weighted behavior cloning:

$$\mathcal{L}_\theta = -\mathbb{E}_{(s,a) \sim D} \left[w_{\nu_\psi, \mu}^*(s, a) \cdot \log \pi_\theta(a \mid s) \right]$$

- 6: **end while**
-

Table 3: Implementation Details for MO Environments

Hyperparameter	Value
β	{1.0, 0.1, 0.01, 0.001, 0.0001}
Hidden dim of ν_ψ and π_θ	768 (512 for MO-Ant)
n_layer of ν_ψ and π_θ	3 (4 for MO-Hopper-3obj)
Learning rate	3×10^{-4}
γ (discount factor)	0.99
Optimizer	Adam

Values in parentheses indicate environment-specific overrides.

922 $\beta \in \{10.0, 1.0, 0.1, 0.01, 0.001, 0.0001, 0.00001\}$. We show that the trade-off between NSW and
 923 distributional shift observed in FairDICE in the finite domain (Figure 2) generally extends to the
 924 continuous domain. NSW performance typically improves as β decreases, reflecting a stronger
 925 emphasis on maximizing NSW and reduced reliance on the dataset distribution. While FairDICE
 926 displays strong NSW across a wide range of β , excessive distributional shift at very small β can
 927 degrade its practical performance.

928 An exception is observed in the MO-Swimmer environment, where NSW consistently decreases as β
 929 decreases. This is likely because the MO-Swimmer dataset already contains trajectories with high
 930 NSW, making further deviation harmful. This is supported by Figure 4, which shows that although
 931 trajectories were generated using different preference weights in existing preference-based baselines,
 932 they consistently achieve high NSW.

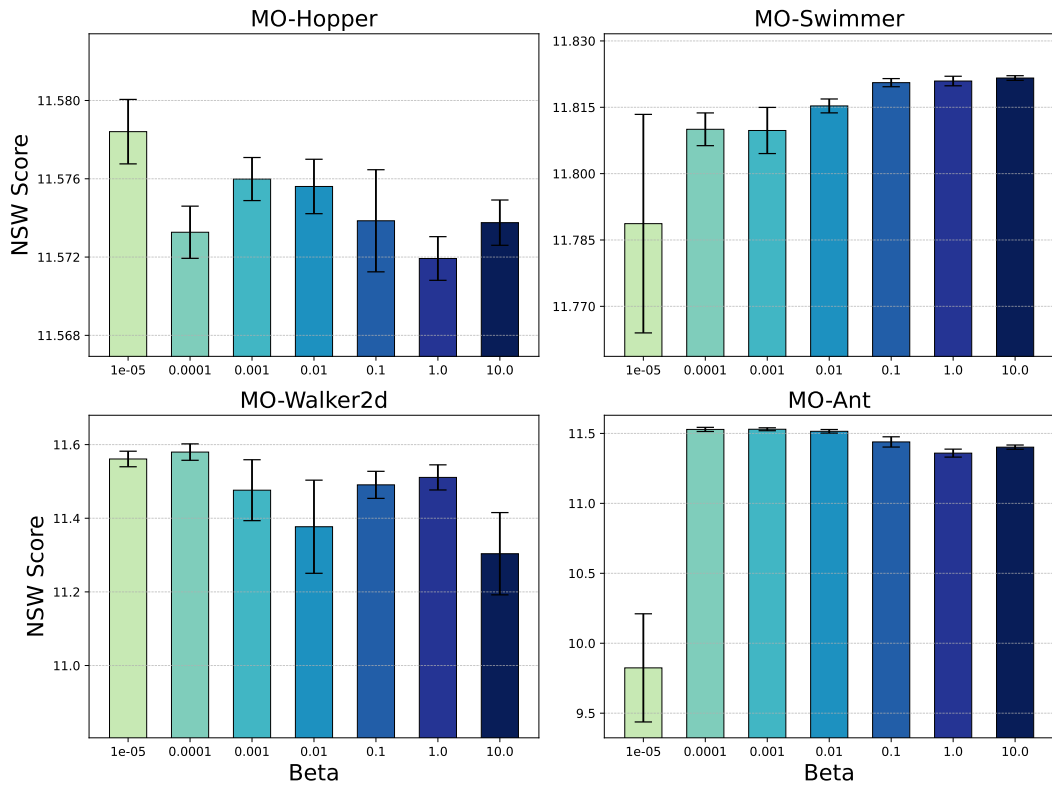


Figure 8: Performance of FairDICE with varying β values on amateur datasets in D4MORL. Results are averaged over 10 seeds and 10 episodes, with error bars denoting ± 1 standard errors.