Multi-objective Reinforcement Learning with Nonlinear Preferences: Provable Approximation for Maximizing Expected Scalarized Return

Nianli Peng*
Harvard University
Cambridge, MA, USA
nianli_peng@g.harvard.edu

Muhang Tian*
Duke University
Durham, NC, USA
muhang.tian@duke.edu

Brandon Fain
Duke University
Durham, NC, USA
btfain@cs.duke.edu

ABSTRACT

We study multi-objective reinforcement learning with nonlinear preferences over trajectories. That is, we maximize the expected value of a nonlinear function over accumulated rewards (expected scalarized return or ESR) in a multi-objective Markov Decision Process (MOMDP). We derive an extended form of Bellman optimality for nonlinear optimization that explicitly considers time and current accumulated reward. Using this formulation, we describe an approximation algorithm for computing an approximately optimal non-stationary policy in pseudopolynomial time for smooth scalarization functions with a constant number of rewards. We prove the approximation analytically and demonstrate the algorithm experimentally, showing that there can be a substantial gap between the optimal policy computed by our algorithm and alternative baselines.

KEYWORDS

Multi-objective Reinforcement Learning, Nonlinear Optimization, Algorithmic Fairness, Approximation Algorithms

ACM Reference Format:
Nianli Peng*, Muhang Tian*, and Brandon Fain. 2025. Multi-objective Reinforcement Learning with Nonlinear Preferences: Provable Approximation for Maximizing Expected Scalarized Return. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

Markov Decision Processes (MDPs) model goal-driven interaction with a stochastic environment, typically aiming to maximize a scalar-valued reward per time-step through a learned policy. Equivalently, this formulation asks the agent to maximize the expected value of a *linear* function of total reward. This problem can be solved with provable approximation guarantees in polynomial time with respect to the size of the MDP [2, 5, 6, 12, 20].

We extend this framework to optimize a *nonlinear* function of vector-valued rewards in multi-objective MDPs (MOMDPs), aiming to maximize expected *welfare* $\mathbb{E}[W(\mathbf{r})]$ where $\mathbf{r}$ is a total reward vector for d objectives. We note that this function is also called the *utility* or the *scalarization* function within the multi-objective optimization literature [18]. Unlike the deep neural networks commonly used as function approximators for large state spaces, our nonlinearity lies entirely in the objective function.

Motivation. Nonlinear welfare functions capture richer preferences for agents, such as fairness or risk attitudes. For example, the Nash Social Welfare function $\left(W_{\text{Nash}}(\mathbf{r}) = (\prod_i r_i)^{1/d}\right)$ reflects a desire for fairness or balance across objectives and diminishing marginal returns [14]. Even single-objective decision theory uses nonlinear utility functions to model risk preferences, like risk aversion with the Von Neumann-Morgenstern utility function [34].

Consider a minimal example with an autonomous taxi robot, Robbie, serving rides in neighborhoods A and B, as diagrammed in Figure 1. Each ride yields a reward in its respective dimension.

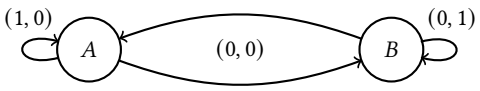


Figure 1: Taxi Optimization Example

Suppose Robbie starts in neighborhood A and has $t = 3$ time intervals remaining before recharging. With no discounting, the *Pareto Frontier* of maximal (that is, undominated) policies can achieve cumulative reward vectors of $(3, 0)$, $(1, 1)$, or $(0, 2)$ by serving A alone, A and B, or B alone respectively. If we want Robbie to prefer the second more balanced or "fair" option of serving one ride in each of the two neighborhoods then Robbie must have *nonlinear* preferences. That is, for any choice of weights on the first and second objective, the simple weighted average would prefer outcomes (3, 0) or (0, 2). However, the second option would maximize the Nash Social Welfare of cumulative reward, for example.

Nonlinear preferences complicate policy computation: Bellman optimality fails, and stationary policies may be suboptimal. Intuitively, a learning agent with fairness-oriented preferences to balance objectives should behave differently, even in the same state, depending on which dimension of reward is "worse off." In the Figure 1 example, one policy to achieve the balanced objective $(1, 1)$ is to complete a ride in A, then travel from A to B, and finally to complete a ride in B – note that this is not stationary with respect to the environment states.

Contributions. In this work, we ask whether it is possible to describe an approximation algorithm (that is, with provable guarantees to approximate the welfare optimal policy) for MOMDPs with nonlinear preferences that has polynomial dependence on the

*Equal contribution.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

number of states and actions, as is the case for linear preferences or scalar MDPs. To the best of our knowledge, ours is the first work to give provable guarantees for this problem, compared to other work that focuses on empirical evaluation of various neural network architectures.

We show this *is* possible for smooth preferences and a constant number of dimensions of reward. To accomplish this, we (i) derive an extended form of Bellman optimality (which may be of independent interest) that characterizes optimal policies for nonlinear preferences over multiple objectives, (ii) describe an algorithm for computing approximately optimal non-stationary policies, (iii) prove the worst-case approximation properties of our algorithm, and (iv) demonstrate empirically that our algorithm can be used in large state spaces to find policies that significantly outperform other baselines.

For additional proofs and experimental details, refer to the appendix in the full version of this paper available at arXiv (https://arxiv.org/abs/2311.02544) and our code is available on GitHub (https://github.com/MuhangTian/Reward-Aware-Value-Iteration).

2 RELATED WORK

Most reinforcement learning algorithms focus on a single scalar-valued objective and maximizing total expected reward [30]. Classic results on provable approximation and runtime guarantees for reinforcement learning include the E3 algorithm [20]. This result showed that general MDPs could be solved to near-optimality efficiently, meaning in time bounded by a polynomial in the size of the MDP (number of states and actions) and the horizon time. Subsequent results refined the achievable bounds [5, 6, 12]. We extend these results to the multi-objective case with nonlinear preferences.

Multi-objective reinforcement learning optimizes multiple objectives at once. So-called single-policy methods use a scalarization function to reduce the problem to scalar optimization for a single policy, and we follow this line of research. The simplest form is linear scalarization, applying a weighted sum to the Q vector [1, 4, 22, 23, 35].

A more general problem is to optimize the expected value of a potentially nonlinear function of the total reward, which may be vector-valued in a multi-objective optimization context. We refer to such a function as a welfare function [7, 16, 28], which is also commonly referred to as a utility or scalarization function [3, 18]. Recent works have explored nonlinear objectives; however, to our knowledge, ours is the first to provide an approximation algorithm with provable guarantees (on the approximation factor), for the expected welfare, leveraging a characterization of recursive optimality in this setting. Several other studies focus on algorithms that demonstrate desirable convergence properties and strong empirical performance by conditioning function approximators on accumulated reward, but without offering approximation guarantees [13, 16, 26, 28]. Complementary to our approach, [25] uses Pareto Conditioned Networks to learn policies for Pareto-optimal solutions by conditioning policies on a preference vector. [21] presents an offline adaptation framework that employs demonstrations to implicitly learn preferences and safety constraints, aligning policies with inferred preferences rather than providing theoretical guarantees.

[3] describe another model-based algorithm that can compute approximately optimal policies for a general class of monotone and Lipschitz welfare functions, but rather than maximizing the expected welfare, they maximize the welfare of expected rewards (note the two are not equal for nonlinear welfare functions). Other works have formulated fairness in different ways or settings. For example, [19] defines an analogue of envy freeness and [15] studies a per-time-step fairness guarantee. [7] studies welfare maximization in multi-armed bandit problems. [27] explores the concept of distributional multi-objective decision making for managing uncertainty in multi-objective environments.

Risk-sensitive RL approaches address scalar objectives by incorporating risk measures to minimize regret or control reward variance over accumulated rewards [9–11]. While these approaches offer valuable tools for managing reward variability, their guarantees are primarily in terms of regret minimization or achieving bounded variance. In contrast, our work provides stronger theoretical assurances in terms of the approximation ratio on the expected welfare for multi-objective optimization. This difference in focus underscores the robustness of our method, which provides guarantees that extend beyond the risk-sensitive regime to cover complex, multi-dimensional utility functions [6, 12, 16]. Such problems remain computationally significant even in a deterministic environment where the notion of risk may not apply.

Lastly, while our single-agent setup with multiple objectives shares some aspects with multi-agent reinforcement learning (MARL), the objectives differ significantly. Much of the MARL literature has focused on cooperative reward settings, often using value-decomposition techniques like VDN and QMIX [24, 29] or actor-critic frameworks to align agent objectives under a centralized training and decentralized execution paradigm [17]. In contrast, our work parallels the more general Markov game setting, where each agent has a unique reward function and studies nonlinear objectives that require computational methods beyond linear welfare functions often assumed in MARL. While MARL research frequently uses linear summations of agent rewards, we demonstrate approximation guarantees for optimizing general, non-linear functions of multiple reward vectors, a distinct contribution in a single-agent setting [3, 19, 34].

3 PRELIMINARIES

A finite Multi-objective Markov Decision Process (MOMDP) consists of a finite set $\mathcal{S}$ of states, a starting state $s_1 \in \mathcal{S}$,* a finite set $\mathcal{A}$ of actions, and probabilities $Pr(s'|s,a) \in [0,1]$ that determine the probability of transitioning to state s' from state s after taking action a. Probabilities are normalized so that $\sum_{s'} Pr(s'|s,a) = 1$.

We have a finite vector-valued reward function $\mathbf{R}(s,a) : \mathcal{S}\times\mathcal{A} \to [0,1]^d$. Each of the d dimensions of the reward vector corresponds to one of the multiple objectives that are to be maximized. At each time step t, the agent observes state $s_t \in \mathcal{S}$, takes action $a_t \in \mathcal{A}(s_t)$, and receives a reward vector $\mathbf{R}(s_t,a_t) \in [0,1]^d$. The environment, in turn, transitions to s_{t+1} with probability $Pr(s_{t+1}|a_t,s_t)$.

To make the optimization objective well-posed in MOMDPs with vector-valued rewards, we must specify a *scalarization* function

*In general we may have a distribution over starting states; we assume a single starting state for ease of exposition.

[18] which we denote as $W : \mathbb{R}^d \to \mathbb{R}$. For fair multi-objective reinforcement learning, we think of each of the d dimensions of the reward vector as corresponding to distinct users. The scalarization function can thus be thought of as a *welfare* function over the users, and the learning agent is a welfare maximizer. Even when $d = 1$, a nonlinear function W can be a Von Neumann-Morgenstern utility function [34] that expresses the risk-attitudes of the learning agent, with strictly concave functions expressing risk aversion.

Assumptions. Here we clarify some preliminary assumptions we make about the reward function $\mathbf{R}(s, a) : \mathcal{S} \times \mathcal{A} \to [0, 1]^d$ and the welfare function $W : \mathbb{R}^d \to \mathbb{R}$. The restriction to $[0, 1]$ is simply a normalization for ease of exposition; the substantial assumption is that the rewards are finite and bounded. Note from the writing we assume the reward function is deterministic: While it suffices for linear optimization to simply learn the mean of random reward distributions, this does not hold when optimizing the expected value of a nonlinear function. Nevertheless, the environment itself may still be stochastic, as the state transitions may still be random.

We also assume that W is **smooth**. For convenience of analysis, we will assume W is **uniformly continuous** on the L1-norm (other parameterizations such as the stronger Lipshitz continuity or using the L2-norm are also possible). For all $\epsilon > 0$, there exists δ_ϵ such that for all $\mathbf{x}, \mathbf{y} \geq 0$,

$$|W(\mathbf{x}) - W(\mathbf{y})| < \epsilon \quad \text{if} \quad \|\mathbf{x} - \mathbf{y}\|_1 < \delta_\epsilon.$$

The smoothness assumption seems necessary to give worst-case approximation guarantees as otherwise arbitrarily small changes in accumulated reward could have arbitrarily large differences in welfare. However, we note that this is still significantly more general than linear scalarization which is implicitly smooth. Practically speaking, our algorithms can be run regardless of the assumed level of smoothness; a particular smoothness is necessary just for the worst-case analysis.

4 MODELING OPTIMALITY

In this section, we expand the classic model of reinforcement learning to optimize the expected value of a nonlinear function of (possibly) multiple dimensions of reward. We begin with the notion of a trajectory of state-action pairs.

Definition 1. *Let M be an MOMDP. A length T trajectory in M is a tuple τ of T state-action pairs*

$$(s_1, a_1), (s_2, a_2), \ldots, (s_T, a_T).$$

For $1 \leq k \leq k' \leq T$, let $\tau_{k:k'}$ be the sub-trajectory consisting of pairs $(s_k, a_k), \ldots, (s_{k'}, a_{k'})$. Let $\tau_{0;0}$ denote the empty trajectory.

For a *discount factor* γ, we calculate the total discounted reward of a trajectory. Note that this is a vector in general.

Definition 2. *For length T trajectory τ and discount factor $0 \leq \gamma \leq 1$, the total discounted reward along τ is the vector*

$$\mathbf{R}(\tau, \gamma) = \sum_{t=1}^{T} \gamma^{t-1} \mathbf{R}(s_t, a_t).$$

For ease of exposition we will frequently leave γ implicit from context and simply write $\mathbf{R}(\tau)$. [†]

A *policy* is a function $\pi(a \mid \tau, s) \in [0, 1]$ mapping past trajectories and current states to probability distributions over actions, that is, $\sum_a \pi(a \mid \tau, s) = 1$ for all τ and s. A *stationary policy* is the special case of a policy that depends only on the current state: $\pi(a \mid s)$.

Definition 3. *The probability that a T-trajectory τ is traversed in an MOMDP M upon starting in state s_1 and executing policy π is*

$$Pr^\pi[\tau] = \pi(a_1 \mid \tau_{0:0}, s_1) \times \prod_{t=2}^{T} \pi\left(a_t \,\middle|\, \tau_{1:t-1}, s_t\right) Pr(s_t | s_{t-1}, a_{t-1}).$$

Problem Formulation. Given a policy, a finite time-horizon T, and a starting state s_1 we can calculate the expected welfare of total discounted reward along a trajectory as follows. Our goal is to maximize this quantity. That is, we want to compute a policy that maximizes the expected T-step discounted welfare.

Definition 4. *For a policy π and a start state s, the expected T-step discounted welfare is*

$$\mathbb{E}_{\tau \sim \pi}\left[W(\mathbf{R}(\tau))\right] = \sum_{\tau} Pr^\pi[\tau] W(\mathbf{R}(\tau))$$

where the expectation is taken over all length T trajectories beginning at s.

Note that this objective is not equal to $W\left(\mathbb{E}_{\tau \sim \pi}\left[\mathbf{R}(\tau)\right]\right)$, which others have studied [3, 28], for a nonlinear W. The former (our objective) is also known as *expected scalarized return* (ESR) whereas the latter is also known as *scalarized expected return* (SER) [18]. While SER makes sense for a repeated decision-making problem, it does not optimize for expected welfare for any particular trajectory. For concave W, ESR $\leq$ SER by Jensen's inequality. However, an algorithm for approximating SER does not provide any guarantee for approximating ESR. For example, a policy can be optimal on SER but achieve 0 ESR if it achieves high reward on one or the other of two objectives but never both in the same episode.

Form of Optimal Policy and Value Functions. The optimal policy for this finite time horizon setting is a function also of the number of time steps remaining. We write such a policy as $\pi(a \mid s, \tau, t)$ where τ is a trajectory (the history), s is the current state, and t is the number of time steps remaining, *i.e.* $t = T - |\tau|$.

We can similarly write the extended value function of a policy π. We write τ as the history or trajectory prior to some current state s and τ' as the future, the remaining t steps determined by the policy π and the environmental transitions.

Definition 5. *The value of a policy π beginning at state s after history τ and with t more actions is*

$$V^\pi(s, \tau, t) = \mathbb{E}_{\tau' \sim \pi}\left[W(\mathbf{R}(\tau) + \gamma^{T-t} \mathbf{R}(\tau'))\right]$$

where the expectation is taken over all length t trajectories τ' beginning at s. The optimal value function is

$$V^*(s, \tau, t) = \max_\pi V^\pi(s, \tau, t)$$

[†] $\gamma < 1$ is necessary for the infinite horizon setting. In the experiments with a finite-horizon task we use $\gamma = 1$ for simplicity.

and the optimal policy is

$$\pi^* \in \arg\max_{\pi} V^{\pi}(s, \tau, t).$$

Note that because W is nonlinear, the value function V^{π} cannot be decomposed in the same way as in the traditional Bellman equations. Before proceeding we want to provide some intuition for this point. The same reasoning helps to explain why stationary policies are not generally optimal.

Suppose we are optimizing the product of the reward between two objectives (i.e., the welfare function is the product or geometric mean), and at some state s with some prior history τ we can choose between two policies π_1 or π_2. Suppose that the future discounted reward vector under π_1 is $(1, 1)$, whereas it is $(10, 0)$ under π_2. So π_1 has greater expected future welfare and traditional Bellman optimality would suggest we should choose π_1. However, if $\mathbf{R}(\tau) = (0, 10)$, we would actually be better off in terms of total welfare choosing π_2. In other words, both past and future reward are relevant when optimizing for the expected value of a nonlinear welfare function.

We develop an extended form of Bellman optimality capturing this intuition by showing that the optimal value function can be written as a function of current state s, accumulated discounted reward $\mathbf{R}(\tau)$, and number of timesteps remaining in the task t. The proof is included in the appendix. At a high level as a sketch, the argument proceeds inductively on t where the base case follows by definition and the inductive step hinges on showing that the the expectation over future trajectories can be decomposed into an expectation over successor states and subsequent trajectories despite the nonlinear W.

Lemma 1. *Let $\mathcal{V}(s, \mathbf{R}(\tau), 0) = W(\mathbf{R}(\tau))$ for all states s and trajectories τ. For every state s, history τ, and $t > 0$ time steps remaining, let*

$$\mathcal{V}(s, \mathbf{R}(\tau), t) = \max_{a} \mathbb{E}_{s'}\left[\mathcal{V}(s', \mathbf{R}(\tau) + \gamma^{T-t}\mathbf{R}(s, a), t-1)\right].$$

Then $V^*(s, \tau, t) = \mathcal{V}(s, \mathbf{R}(\tau), t)$.

By Lemma 1, we can parameterize the optimal value function by the current state s, accumulated reward $\mathbf{R}_{\mathbf{acc}} \in \mathbb{R}^d$, and number of timesteps remaining t. We will use this formulation of V^* in the remainder of the paper.

Definition 6 (Recursive Formulation of V^*). *Let $\mathbf{R}_{\mathbf{acc}} = \mathbf{R}(\tau)$ be the vector of accumulated reward along a history prior to some state s. Let $V^*(s, \mathbf{R}_{\mathbf{acc}}, 0) = W(\mathbf{R}_{\mathbf{acc}})$ for all s and $\mathbf{R}_{\mathbf{acc}}$. For $t > 0$,*

$$V^*(s, \mathbf{R}_{acc}, t) = \max_{a} \sum_{s'} Pr(s'|s, a) \cdot V^*(s', \mathbf{R}_{acc} + \gamma^{T-t}\mathbf{R}(s, a), t-1).$$

Horizon Time. Note that an approximation algorithm for this discounted finite time horizon problem can also be used as an approximation algorithm for the discounted infinite time horizon problem. Informally, discounting by γ with bounded maximum reward implies that the first $T \approx 1/(1-\gamma)$ steps dominate overall returns. We defer the precise formulation of the lower bound on the horizon time and its proof in the appendix (Lemma **??**).

Necessity of Conditioning on Remaining Timesteps t. We illustrate the necessity of conditioning the optimal value function on the remaining timesteps t using a counterexample in the appendix.

5 COMPUTING OPTIMAL POLICIES

Our overall algorithm is Reward-Aware Explore or Exploit (or RAEE for short) to compute an approximately optimal policy, inspired by the classical E3 algorithm [20]. At a high level, the algorithm explores to learn a model of the environment, periodically pausing to recompute an approximately optimal policy on subset of the environment that has been thoroughly explored. We call this optimization subroutine Reward-Aware Value Iteration (or RAVI for short). In both cases, reward-aware refers to the fact that the algorithms compute non-stationary policies in the sense that optimal behavior depends on currently accumulated vector-valued reward, in addition to the current state.

Of the two, the model-based optimization subroutine RAVI is the more significant. The integrated model-learning algorithm RAEE largely follows from prior work, given access to the RAVI subroutine. For this reason, we focus in this section on RAVI, and defer a more complete discussion and analysis of RAEE to Section 7 and the appendix.

A naive algorithm for computing a non-stationary policy would need to consider all possible prior trajectories for each decision point, leading to a runtime complexity containing the term $|\mathcal{S}|^T$, exponential in the size of the state space and time horizon. Instead, for a smooth welfare function on a constant number of objectives, our algorithm will avoid any exponential dependence on $|\mathcal{S}|$.

The algorithm is derived from the recursive definition of the optimal multi-objective value function V^* in Definition 6, justified by Lemma 1, parameterized by the accumulated discounted reward vector $\mathbf{R}_{\mathbf{acc}}$ instead of the prior history. Note that even if the rewards were integers, $\mathbf{R}_{\mathbf{acc}}$ might not be due to discounting. We must therefore introduce a discretization that maps accumulated reward vectors to points on a lattice, parameterized by some $\alpha \in (0, 1)$ where a smaller α leads to a finer discretization but increases the runtime.

Definition 7. *For a given discretization precision parameter $\alpha \in \mathbb{R}^+$, define $f_\alpha : \mathbb{R}^d \to (\alpha\mathbb{Z})^d$ by*

$$f_\alpha(\mathbf{R}) = \left(\left\lfloor \frac{R_1}{\alpha} \right\rfloor \cdot \alpha, \left\lfloor \frac{R_2}{\alpha} \right\rfloor \cdot \alpha, \cdots, \left\lfloor \frac{R_d}{\alpha} \right\rfloor \cdot \alpha\right).$$

In other words, f_α maps any d-dimensional vector to the largest vector in $(\alpha\mathbb{Z})^d$ that is less than or equal to the input vector, effectively rounding each component *down* to the nearest multiple of α. We now describe the algorithm, which at a high level computes the dynamic program of approximately welfare-optimal value functions conditioned on discretized accumulated reward vectors.

Remark. Since we model the per-step reward as normalized to at most 1, we describe α as lying within $(0, 1)$. However, the algorithm itself is well-defined for larger values of alpha (coarser than the per-step reward) and during implementation and experiments, we consider such larger α, beyond what might give worst-case guarantees but still observing strong empirical performance.

The asymptotic runtime complexity is $O\left(|\mathcal{S}|^2|\mathcal{A}|(T/\alpha)^d\right)$. However, observe that the resulting algorithm is extremely parallelizable: Given solutions to subproblems at $t-1$, all subproblems for t can in principle be computed in parallel. A parallel implementation of RAVI can leverage GPU compute to handle the extensive calculations involved in multi-objective value iteration. Each thread

Algorithm 1 Reward-Aware Value Iteration (RAVI)

Parameters: Discretization precision $\alpha \in (0,1)$, Discount factor $\gamma \in [0,1)$, Reward dimension d, finite time horizon T, welfare function W, discretization function f_α.
Require: Normalize $\mathbf{R}(s,a) \in [0,1]^d$ for all $s \in \mathcal{S}, a \in \mathcal{A}$.
Initialize: $V(s, \mathbf{R_{acc}}, 0) = W(\mathbf{R_{acc}})$ for all $\mathbf{R_{acc}} \in \{0, \alpha, 2\alpha, \ldots, \lceil T/\alpha \rceil \cdot \alpha\}^d$, $s \in \mathcal{S}$.
for t = 1 to T **do**
  for $\mathbf{R_{acc}} \in \{0, \alpha, 2\alpha, \ldots, \lceil (T-t)/\alpha \rceil \cdot \alpha\}^d$ **do**
    for all $s \in \mathcal{S}$ **do**

$$V\left(s, \mathbf{R_{acc}}, t\right) \leftarrow \max_a \sum_{s'} Pr\left(s'|s,a\right) V\left(s', \varphi(s, \mathbf{R}_{acc}, a), t-1\right)$$

$$\pi\left(s, \mathbf{R_{acc}}, t\right) \leftarrow \arg\max_a \sum_{s'} Pr\left(s'|s,a\right) V\left(s', \varphi(s, \mathbf{R}_{acc}, a), t-1\right)$$

where $\varphi(s, \mathbf{R}_{acc}, a) := f_\alpha\left(\mathbf{R}_{acc} + \gamma^{T-t}\mathbf{R}(s,a)\right)$

computes the value updates and policy decisions for a specific state and accumulated reward combination, which allows for massive parallelism. In practice, we observed an empirical speedup of approximately 560 times on a single NVIDIA A100 compared to the CPU implementation on an AMD Ryzen 9 7950x 16-core processor for our experimental settings. This drastic improvement in runtime efficiency makes it feasible to run RAVI on larger and more complex environments, where the computational demands would otherwise be prohibitive.

It remains to see how small α needs to be, which will dictate the final runtime complexity. We first analyze the correctness of the algorithm and then return to the setting of α for a given smoothness of W to conclude the runtime analysis.

We begin analyzing the approximation of RAVI by showing an important structural property: the optimal value function will be smooth as long as the welfare function is smooth. The proof is included in the appendix.

Lemma 2 (Uniform continuity of multi-objective value function). *Let the welfare function $W : \mathbb{R}^d \to \mathbb{R}$ be uniformly continuous. Fix $s \in \mathcal{S}$ and $t \in \{0, 1, \ldots, T\}$, then for all $\epsilon > 0$, there exists $\delta_\epsilon > 0$ such that*

$$\left|V^*(s, \mathbf{R_1}, t) - V^*(s, \mathbf{R_2}, t)\right| < \epsilon \quad \text{if} \quad \|\mathbf{R_1} - \mathbf{R_2}\| < \delta_\epsilon.$$

We now present the approximation guarantee of RAVI, that it achieves an additive error that scales with the smoothness of W and the number of remaining time steps. The proof of this lemma is deferred to the appendix.

Lemma 3 (Approximation Error of RAVI). *For uniformly continuous welfare function W, for all $\epsilon > 0$, there exists α_ϵ such that*

$$V(s, \mathbf{R_{acc}}, t) \geq V^*(s, \mathbf{R_{acc}}, t) - t\epsilon$$

$\forall s \in \mathcal{S}, \mathbf{R_{acc}} \in \mathbb{R}^d, t \in \{0, 1, \ldots, T\}$, where $V(s, \mathbf{R_{acc}}, t)$ is computed by Algorithm 1 using α_ϵ.

While we can use any setting of α empirically, this shows that for an approximation guarantee we should set the discretization parameter to $\alpha = \delta_{T\epsilon}/d$ where $\delta_{T\epsilon}$ from the smoothness of the welfare function is sufficient to drive its bound to ϵ, that is, it should be $\alpha = \frac{\delta_\epsilon}{T \cdot d}$.

We thus arrive at the ultimate statement of the approximation and runtime of RAVI. The proof follows directly from Lemma 3 and the setting of α.

Theorem 1 (Optimality Guarantee of RAVI). *For a given ϵ and welfare function W that is δ_ϵ uniformly continuous, RAVI with $\alpha = \frac{\delta_\epsilon}{T \cdot d}$ computes a policy $\hat{\pi}$, such that $V^{\hat{\pi}}(s, 0, T) \geq V^*(s, 0, T) - \epsilon$ in $O\left(|\mathcal{S}|^2 |\mathcal{A}| (d \cdot T^2/\delta_\epsilon)^d\right)$ time.*

A concrete example of a particular welfare function, the setting of relevant parameters, and a derivation of a simplified runtime statement may help to clarify Theorem 1. Consider the smoothed *proportional fairness* objective: $W_{\text{SPF}}(\mathbf{x}) = \sum_{i=1}^d \ln(x_i + 1)$, a smoothed log-transform of the Nash Social Welfare (or geometric mean) with better numerical stability.

By taking the gradient, we can see that $W_{\text{SPF}}(\mathbf{x})$ is d-Lipschitz on $\mathbf{x} \geq 0$, so we may pick $\delta_\epsilon = \epsilon/d$ to satisfy the uniform continuity requirement in Theorem 1. ‡

Plugging δ_ϵ into the runtime and recalling that $\alpha_{T\epsilon} = \frac{\delta_{T\epsilon}}{d}$ and $\delta_{T\epsilon} = \delta_\epsilon/T$ up to constant factors, we get

$$O\left(|\mathcal{S}|^2 |\mathcal{A}| (T/\alpha_{T\epsilon})^d\right) = O\left(|\mathcal{S}|^2 |\mathcal{A}| \left(\frac{T^2 \cdot d^2}{\epsilon}\right)^d\right).$$

To further simplify, if one takes the number of actions $|\mathcal{A}|$, discount factor γ, and dimension d to be constants, then the asymptotic dependence of the runtime (in our running example) is

$$O\left(|\mathcal{S}|^2 \left(\frac{1}{\epsilon} \log^2(1/\epsilon)\right)^d\right).$$

This dependence is significantly better than a naive brute-force approach, which scales at least as $O(|\mathcal{S}|^T)$.

Intuitively, the key savings arise from discretizing the reward space (with granularity δ_ϵ) rather than enumerating all possible trajectories. This discretization is guided by the smoothness assumption on W_{SPF}.

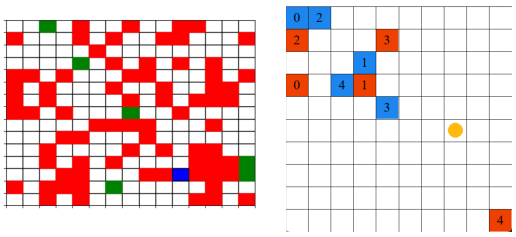
6 EXPERIMENTS

We test RAVI on two distinct interpretations of multi-objective reinforcement learning: (1) the fairness interpretation, where the agent tries to maximize rewards across all dimensions. (2) the objective interpretation, where the agent tries to maximize one while minimizing the other. In both scenarios, we show that RAVI can discover more optimal policies than other baselines for nonlinear multi-objective optimization. This holds even for coarser settings of α in the algorithm than would be necessary for strong theoretical worst-case approximation guarantees.

6.1 Simulation Environments

Taxi: We use the taxi multi-objective simulation environment considered in [16] for testing nonlinear ESR maximization. In this

‡Uniform continuity is a weaker modeling assumption than L-Lipshitz continuity for constant L. Note that for an L-Lipshitz function, the correct value of δ_ϵ is just ϵ/L, where ϵ is the desired approximation factor of the algorithm.



Scavenger Taxi, $d = 5$

Figure 2: Visualization of the Taxi and Scavenger environments.

grid-world environment, the agent is a taxi driver whose goal is to deliver passengers. There are multiple queues of passengers, each having a given pickup and drop-off location. Each queue has a different objective or dimension of the reward. The taxi can only take one passenger at a time and wants to balance trips for passengers in the different queues. This environment models the fairness interpretation.

Scavenger: Inspired by the Resource Gathering environment [8], the scavenger hunt environment is a grid-world simulation where the agent must collect resources while avoiding enemies scattered across the grid. The state representation includes the agent's position and the status of the resources (collected or not). The reward function is vector-valued, where the first component indicates the number of resources collected, and the second component indicates the damage taken by encountering enemies. This environment is the objective interpretation.

6.2 Baseline Algorithms

Linearly Scalarized Policy (LinScal) [33]: A relatively straightforward technique for multi-objective RL optimization is to apply the linear combination on the Q-values for each objective. Given weights $\mathbf{w} \in \mathbb{R}^d$, $\sum_i^d w_i = 1$, the scalarized objective is $SQ(s,a) = \mathbf{w}^\top \mathbf{Q}(s,a)$, where $Q(s,a)_i$ is the Q-value for the i^{th} objective. ϵ-greedy policy is used on $SQ(s,a)$ during action selection, and regular Q-learning updates are applied on each reward dimension.

Mixture Policy (Mixture) [32]: this baseline works by combining multiple Pareto optimal base policies into a single policy. Q-learning is used to optimize for each reward dimension separately (which is approximately Pareto optimal), and the close-to-optimal policy for each dimension is used for I steps before switching to the next.

Welfare Q-learning (WelfareQ) [16]: this baseline extends Q-learning in tabular setting to approximately solve the nonlinear objective function by considering past accumulated rewards to perform non-stationary action selection.

Model-Based Mixture Policy (Mixture-M): Instead of using Q-learning to find an approximately Pareto optimal policy for each objective, value iteration [31] is used to calculate the optimal policy, and each dimension uses the corresponding optimal policy for I steps before switching to the next.

6.3 Nonlinear Functions

Taxi: We use the following three functions on Taxi for fairness considerations: (1) Nash social welfare: $W_{\text{Nash}}(\mathbf{r}) = (\prod_i r_i)^{1/d}$. (2) Egalitarian welfare: $W_{\text{Egalitarian}}(\mathbf{r}) = \min\{r_i\}_i$. (3) p-welfare[§]: $W_p(\mathbf{r}) = (\frac{1}{d}\sum_i r_i^p)^{1/p}$.

Scavenger: We use these two functions to reflect the conflicting nature of the objectives:

(1) Resource-Damage Threshold Scalarization: $RD_{\text{threshold}}(R,D) = R - \max(0, (D - \text{threshold})^3)$, where R is number of resources collected and D is damage taken from enemies. The threshold parameter represents a budget after which the penalty from the damage starts to apply.
(2) Cobb-Douglas Scalarization: $CD_\rho(R,D) = R^\rho\,(1/(D+1))^{(1-\rho)}$. This function is inspired by economic theory and balances trade-offs between R and D.

6.4 Experiment Settings and Hyperparameters

We run all the algorithms using 10 random initializations with a fixed seed each. We set $\mathbf{w} = (1/d) \times \mathbf{1}$ (uniform weight) for LinScal, $I = T/d$ for Mixture and Mixture-M, $\alpha = 1$ for RAVI, and learning rate of 0.1 for WelfareQ. Three of our baselines (Mixture, LinScal, and WelfareQ) are online algorithms. Thus, to ensure a fair comparison, we tuned their hyperparameters using grid-search and evaluated their performances after they reached convergence. For model-based approaches (RAVI and Mixture-M), we run the algorithms and evaluate them after completion. We set the convergence threshold to $\Delta = 10^{-7}$ for Mixture-M. Some environment-specific settings are discussed below.

Taxi: To ensure numerical stability of W_{Nash}, we optimize its smoothed log-transform $W_{\text{SPF}}(\mathbf{r},\lambda) = \sum_{i=1}^d \ln(r_i + \lambda)$, but during evaluations we still use W_{Nash}. We set $\lambda = 10^{-8}$, $T = 100$. $\gamma = 1$, size of the grid world to be 15×15 with $d \in \{2,3,4,5\}$ reward dimensions.

Scavenger: We set threshold for $RD_{\text{threshold}}$ to 2 and $\rho = 0.4$ for CD_ρ. The size of the grid world is 15×15 with six resources scattered randomly and 1/3 of the cells randomly populated with enemies. We use $T = 20$ and $\gamma = 1$.

6.5 Results

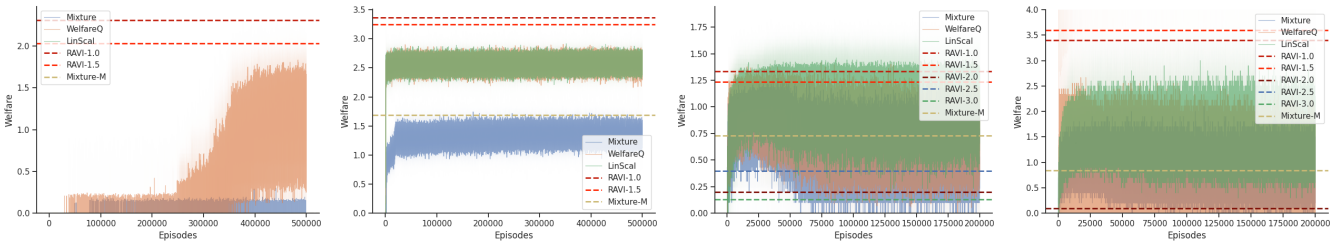
As shown in Table 1, we found that RAVI is able to generally outperform all baselines across all settings in both Taxi and Scavenger environments in terms of optimizing our nonlinear functions of interest.

On the Taxi environment, we observe that LinScal is unable to achieve any performance except for $W_{p=0.9}$. This is an expected behavior due to the use of a linearly scalarized policy and the fact that p-welfare converges to utilitarian welfare when $p \to 1$. Among the baseline algorithms, we observe that WelfareQ performs the best on $d = 5$, whereas Mixture and Mixture-M fail due to the use of a fixed interval length for each optimal policy. Furthermore, the advantage of RAVI becomes more obvious as d increases.

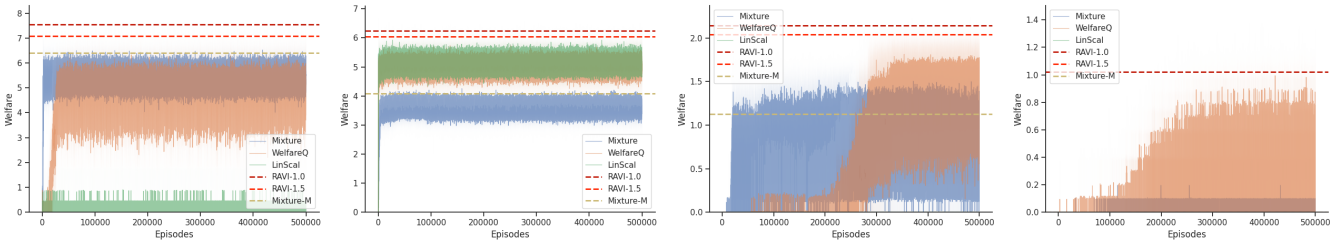
[§] p-welfare is equivalent to generalized mean. If $p \to 0$, p-welfare converges to Nash welfare; if $p \to -\infty$, p-welfare converges to egalitarian welfare; and p-welfare is the utilitarian welfare when $p = 1$.

Table 1: Comparison with baselines in terms of ESR.

Environment	Dimension	Function	RAVI	Mixture	LinScal	WelfareQ	Mixture-M
Taxi	$d = 2$	W_{Nash}	**7.555±0.502**	5.136±0.547	0.000±0.000	5.343±0.964	6.406±0.628
		$W_{\text{Egalitarian}}$	3.000±0.000	3.483±0.894	0.000±0.000	0.387±0.475	**4.065±0.773**
		$W_{p=-10}$	**5.279±0.000**	3.623±0.522	0.000±0.000	2.441±1.518	4.356±0.829
		$W_{p=0.001}$	**7.404±0.448**	5.363±0.844	0.000±0.000	4.977±1.514	6.406±0.628
		$W_{p=0.9}$	**9.628±0.349**	5.947±0.488	7.833±0.908	7.999±0.822	7.052±0.412
	$d = 3$	W_{Nash}	**4.996±0.297**	3.250±0.584	0.000±0.000	2.798±1.462	3.461±0.459
		$W_{\text{Egalitarian}}$	2.000±0.000	**2.030±0.981**	0.000±0.000	0.094±0.281	1.660±0.663
		$W_{p=-10}$	**3.115±0.000**	2.129±0.878	0.000±0.000	2.558±0.812	1.849±0.733
		$W_{p=0.001}$	3.307±0.000	3.118±0.536	0.000±0.000	3.306±1.181	**3.462±0.458**
		$W_{p=0.9}$	**6.250±0.322**	3.883±0.329	5.191±0.464	4.834±0.797	4.081±0.266
	$d = 4$	W_{Nash}	**2.191±0.147**	0.579±0.713	0.000±0.000	1.601±0.189	1.122±0.781
		$W_{\text{Egalitarian}}$	**1.700±0.483**	0.545±0.445	0.000±0.000	0.000±0.000	0.634±0.437
		$W_{p=-10}$	**1.029±0.000**	0.490±0.490	0.000±0.000	0.689±0.451	0.688±0.475
		$W_{p=0.001}$	**2.145±0.129**	0.961±0.628	0.000±0.000	1.440±0.511	1.126±0.774
		$W_{p=0.9}$	**3.369±0.173**	1.230±0.264	2.546±0.174	2.521±0.211	1.689±0.320
	$d = 5$	W_{Nash}	**2.308±0.185**	0.000±0.000	0.000±0.000	1.181±0.613	0.000±0.000
		$W_{\text{Egalitarian}}$	**1.700±0.483**	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
		$W_{p=-10}$	**1.023±0.000**	0.000±0.000	0.000±0.000	0.407±0.501	0.000±0.000
		$W_{p=0.001}$	**2.000±0.102**	0.018±0.020	0.000±0.000	1.309±0.490	0.025±0.025
		$W_{p=0.9}$	**3.289±0.147**	1.220±0.141	2.844±0.250	2.879±0.263	1.761±0.148
Scavenger	$d = 2$	$CD_{\rho=0.4}$	**1.336±0.240**	0.655±0.558	0.874±0.605	0.713±0.645	0.729±0.296
		$RD_{\text{threshold}=2}$	**3.400±0.843**	0.900±0.943	1.200±0.872	1.444±1.423	0.845±2.938



Taxi, W_{Nash}**,** $d = 5$ **Taxi,** $W_{p=0.9}$**,** $d = 4$ **Scavenger,** $CD_{\rho=0.4}$ **Scavenger,** $RD_{\text{threshold}}$



Taxi, W_{Nash}**,** $d = 2$ **Taxi,** $W_{p=0.9}$**,** $d = 3$ **Taxi,** $W_{p=0.001}$**,** $d = 4$ **Taxi,** $W_{p=-10}$**,** $d = 5$

Figure 3: Comparisons with baselines, with learning curves included.

On the Scavenger environment, due to $d = 2$ and having a dense reward signal, all algorithms are able to achieve reasonable values for the welfare functions. We also observe that RAVI significantly outperforms all baselines.

Given that Mixture, LinScal, and WelfareQ are online algorithms, for the sake of completeness, we provide the visualizations of the learning curves of the baseline algorithms compared with RAVI with different α values in Figure 3, where the model-base approaches

are shown as horizontal dotted lines. In general, we found that the online algorithms tend to converge early, and we observe a degradation in performance as α increases. The full set of results can be found at **??**.

6.6 Ablation Study on Discretization Factor

To further evaluate the effect of α on RAVI, we also run two sets of experiments:

(1) we test RAVI performance on settings with $\gamma < 1$ and adopt α values that are substantially greater than those necessary for worst-case theoretical guarantees from the previous analysis. This set of results can be found at **??**. Note that the empirical performance of RAVI is substantially better than is guaranteed by the previous theoretical analysis.
(2) With $\gamma = 1$, we evaluate RAVI using larger alpha values and investigate how much performance degradation occurs. This set of experiments can be found in **??**.

7 REMOVING KNOWLEDGE OF THE MODEL

We have shown that RAVI can efficiently find an approximately optimal policy given access to a model of the environment. In this section we observe that the model can be jointly learned by extending the classical E3 algorithm [20] to the nonlinear multi-objective case by lifting the exploration algorithm to the multi-objective setting and using RAVI for the exploitation subroutine. We call the resulting combined algorithm Reward-Aware-Explore or Exploit (or RAEE for short). We briefly explain the high level ideas and state the main result here and defer further discussion to the appendix due to space constraints.

The algorithm consists of two stages, exploration and exploitation. The algorithm alternates between two stages: exploration and exploitation. Each stage is outlined here at a high level, with more detailed steps provided in the appendix.

- **Explore.** At a given state, choose the least experienced action and record the reward and transition. Continue in this fashion until reaching a *known* state, where we say a state is known if it has been visited sufficiently many times for us to have precise local statistics about the reward and transition functions in that state.
- **Exploit.** Run RAVI from the current known state s in the induced model comprising the known states and with a single absorbing state representing all unknown states. If the welfare obtained by this policy is within the desired error bound of $V^*(s, \mathbf{0}, T)$, then we are done. Otherwise, compute a policy to reach an unknown state as quickly as possible and resume exploring.

Theorem 2 (RAEE). *Let $V^*(s, 0, T)$ denote the value function of the policy with the optimal expected welfare in the MOMDP M starting at state s, with $\mathbf{0} \in \mathbb{R}^d$ accumulated reward and T timesteps remaining. Then for a uniformly continuous welfare function W, there exists an algorithm A, taking inputs ϵ, β, $\mathcal{S}$, $\mathcal{A}$, and $V^*(s, \mathbf{0}, T)$, such that the total number of actions and computation time taken by A is polynomial in $1/\epsilon$, $1/\beta$, $|\mathcal{S}|$, $|\mathcal{A}|$, the horizon time $T = 1/(1-\gamma)$ and exponential in the number of objectives d, and with probability at least $1-\beta$, A will halt in a state s, and output a policy $\hat{\pi}$, such that $V_M^{\hat{\pi}}(s, 0, T) \geq V^*(s, 0, T) - \epsilon$.*

We provide additional details comparing our analysis with that of [20] in the appendix.

As we do not regard the learning of the Multi-Objective Markov Decision Process (MOMDP) as our primary contribution, we choose to focus the empirical evaluation on the nonlinear optimization subroutine, which is the most crucial modification from the learning problem with a single objective. The environments we used for testing the optimality of RAVI are very interesting for optimizing a nonlinear function of the rewards, but are deterministic in terms of transitions and rewards, making the learning of these environments less interesting empirically.

8 CONCLUSION AND FUTURE WORK

Nonlinear preferences in reinforcement learning are important, as they can encode *fairness* as the nonlinear balancing of priorities across multiple objectives or *risk attitudes* with respect to even a single objective. Stationary policies are not necessarily optimal for such objectives. We derived an extended form of Bellman optimality to characterize the structure of optimal policies conditional on accumulated reward. We introduced the RAVI and RAEE algorithms to efficiently compute an approximately optimal policy.

Our work is certainly not the first to study MORL including with nonlinear preferences. However, to the best of our knowledge, our work is among the first to provide worst-case approximation guarantees for optimizing ESR in MORL with nonlinear scalarization functions.

While our experiments demonstrate the utility of RAVI in specific settings, there are many possible areas of further empirical evaluation including stochastic environments and model learning alongside the use of RAVI as an exploitation subprocedure as described in theory in Section 7 with RAEE. Further details on the limitations of our experiments are discussed in **??**.

Our results introduce several natural directions for future work. On the technical side, one could try to handle the case of stochastic reward functions or a large number of objectives. Another direction would involve incorporating function approximation with deep neural networks into the algorithms to enable scaling to even larger state spaces and generalizing between experiences. Our theory suggests that it may be possible to greatly enrich the space of possible policies that can be efficiently achieved in these settings by conditioning function approximators on accumulated reward rather than necessarily considering sequence models over arbitrary past trajectories; we see this as the most exciting next step for applications.

between

REFERENCES

[1] Axel Abels, Diederik Roijers, Tom Lenaerts, Ann Nowé, and Denis Steckelmacher. 2019. Dynamic weights in multi-objective deep reinforcement learning. In *International conference on machine learning*. PMLR, 11–20.
[2] Alekh Agarwal, Sham M. Kakade, Jason D. Lee, and Gaurav Mahajan. 2021. On the Theory of Policy Gradient Methods: Optimality, Approximation, and Distribution Shift. *J. Mach. Learn. Res.* 22, 1, Article 98 (jan 2021), 76 pages.
[3] Mridul Agarwal, Vaneet Aggarwal, and Tian Lan. 2022. Multi-Objective Reinforcement Learning with Non-Linear Scalarization. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems* (Virtual Event, New Zealand) *(AAMAS '22)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 9–17.
[4] Lucas N Alegre, Ana LC Bazzan, Diederik M Roijers, Ann Nowé, and Bruno C da Silva. 2023. Sample-efficient multi-objective learning via generalized policy

improvement prioritization. *arXiv preprint arXiv:2301.07784* (2023).
[5] Peter Auer, Thomas Jaksch, and Ronald Ortner. 2008. Near-Optimal Regret Bounds for Reinforcement Learning. In *Proceedings of the 21st International Conference on Neural Information Processing Systems* (Vancouver, British Columbia, Canada) *(NIPS'08)*. Curran Associates Inc., Red Hook, NY, USA, 89–96.
[6] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. 2017. Minimax Regret Bounds for Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 263–272. https://proceedings.mlr.press/v70/azar17a.html
[7] Siddharth Barman, Arindam Khan, Arnab Maiti, and Ayush Sawarni. 2023. Fairness and Welfare Quantification for Regret in Multi-Armed Bandits. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 6 (Jun. 2023), 6762–6769. https://doi.org/10.1609/aaai.v37i6.25829
[8] Leon Barrett and Srini Narayanan. 2008. Learning all optimal policies with multiple criteria. In *Proceedings of the 25th International Conference on Machine Learning* (Helsinki, Finland) *(ICML '08)*. Association for Computing Machinery, New York, NY, USA, 41–47. https://doi.org/10.1145/1390156.1390162
[9] Osbert Bastani, Jason Yinglun Ma, Ethan Shen, and Weiran Xu. 2022. Regret bounds for risk-sensitive reinforcement learning. In *Advances in Neural Information Processing Systems*, Vol. 35. 36259–36269.
[10] Nicole Bäuerle and Jonathan Ott. 2011. Markov decision processes with average-value-at-risk criteria. *Mathematical Methods of Operations Research* 74 (2011), 361–379.
[11] Marc G Bellemare, Will Dabney, and Mark Rowland. 2023. *Distributional Reinforcement Learning*. MIT Press.
[12] Ronen I. Brafman and Moshe Tennenholtz. 2003. R-Max - a General Polynomial Time Algorithm for near-Optimal Reinforcement Learning. *J. Mach. Learn. Res.* 3, null (mar 2003), 213–231. https://doi.org/10.1162/153244303765208377
[13] Xin-Qiang Cai, Pushi Zhang, Li Zhao, Jiang Bian, Masashi Sugiyama, and Ashley Llorens. 2023. Distributional Pareto-Optimal Multi-Objective Reinforcement Learning. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 15593–15613. https://proceedings.neurips.cc/paper_files/paper/2023/file/32285dd184dbfc33cb2d1f0db53c23c5-Paper-Conference.pdf
[14] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D Procaccia, Nisarg Shah, and Junxing Wang. 2019. The unreasonable fairness of maximum Nash welfare. *ACM Transactions on Economics and Computation (TEAC)* 7, 3 (2019), 1–32.
[15] Zhun Deng, He Sun, Steven Wu, Linjun Zhang, and David Parkes. 2023. Reinforcement Learning with Stepwise Fairness Constraints. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 206)*, Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent (Eds.). PMLR, 10594–10618. https://proceedings.mlr.press/v206/deng23a.html
[16] Ziming Fan, Nianli Peng, Muhang Tian, and Brandon Fain. 2023. Welfare and Fairness in Multi-Objective Reinforcement Learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems* (London, United Kingdom) *(AAMAS '23)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1991–1999.
[17] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. *Proceedings of the AAAI Conference on Artificial Intelligence* 32, 1 (2018).
[18] Conor F. Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Kallstrom, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M. Zintgraf, Richard Dazeley, Fredrik Heintz, Enda Howley, Athirai A. Irissappane, Patrick Mannion, Ann Nowé, Gabriel Ramos, Marcello Restelli, Peter Vamplew, and Diederik M. Roijers. 2023. A Brief Guide to Multi-Objective Reinforcement Learning and Planning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems* (London, United Kingdom) *(AAMAS '23)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1988–1990.
[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] Michael Kearns and Satinder Singh. 2002. Near-optimal reinforcement learning in polynomial time. *Machine learning* 49 (2002), 209–232.
[21] Qian Lin, Zongkai Liu, Danying Mo, and Chao Yu. 2024. An Offline Adaptation Framework for Constrained Multi-Objective Reinforcement Learning. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*. https://neurips.cc/virtual/2024/poster/95257
[22] Chunming Liu, Xin Xu, and Dewen Hu. 2014. Multiobjective reinforcement learning: A comprehensive overview. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45, 3 (2014), 385–398.
[23] Kristof Van Moffaert, Madalina M Drugan, and Ann Nowé. 2013. Hypervolume-based multi-objective reinforcement learning. In *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 352–366.
[24] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning*.
[25] M. Reymond, E. Bargiacchi, and A. Nowé. 2022. Pareto Conditioned Networks. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 1110–1118.
[26] Mathieu Reymond, Conor F Hayes, Denis Steckelmacher, Diederik M Roijers, and Ann Nowé. 2023. Actor-critic multi-objective reinforcement learning for non-linear utility functions. *Autonomous Agents and Multi-Agent Systems* 37, 2 (2023), 23.
[27] W. Röpke, C. F. Hayes, P. Mannion, E. Howley, A. Nowé, and D. M. Roijers. 2023. Distributional multi-objective decision making. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. 5711–5719.
[28] Umer Siddique, Paul Weng, and Matthieu Zimmer. 2020. Learning Fair Policies in Multiobjective (Deep) Reinforcement Learning with Average and Discounted Rewards. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. JMLR.org, Article 826, 11 pages.
[29] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech M Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, and Thore Graepel. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296* (2017).
[30] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.
[31] Richard S Sutton, Andrew G Barto, et al. 1998. Introduction to reinforcement learning. (1998).
[32] Peter Vamplew, Richard Dazeley, Ewan Barker, and Andrei Kelarev. 2009. Constructing stochastic mixture policies for episodic multiobjective reinforcement learning tasks. In *Australasian joint conference on artificial intelligence*. Springer, 340–349.
[33] Kristof Van Moffaert, Madalina M Drugan, and Ann Nowé. 2013. Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, 191–199.
[34] John Von Neumann and Oskar Morgenstern. 1947. *Theory of games and economic behavior, 2nd rev.* Princeton university press.
[35] Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. 2019. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. *Advances in neural information processing systems* 32 (2019).