Multi-Objective Reinforcement Learning with Non-Linear Scalarization

Mridul Agarwal
Purdue University
West Lafayette, IN, USA
agarw180@purdue.edu

Vaneet Aggarwal
Purdue University
West Lafayette, IN, USA
vaneet@purdue.edu

Tian Lan
George Washington University
Washington, D.C., USA
tlan@gwu.edu

ABSTRACT

Multi-Objective Reinforcement Learning (MORL) setup naturally arises in many places where an agent optimizes multiple objectives. We consider the problem of MORL where multiple objectives are combined using a non-linear scalarization. We combine the vector objectives with a concave scalarization function and maximize this scalar objective. To work with the non-linear scalarization, in this paper, we propose a solution using steady-state occupancy measures and long-term average rewards. We show that when the scalarization function is element-wise increasing, the optimal policy for the scalarization is also Pareto optimal. To maximize the scalarized objective, we propose a model-based posterior sampling algorithm. Using a novel Bellman error analysis for infinite horizon MDPs based proof, we show that the proposed algorithm obtains a regret bound of $\tilde{O}(LKDS\sqrt{A/T})$ for K objectives, and L-Lipschitz continous scalarization function for MDP with S states, A actions, and diameter D. Additionally, we propose policy-gradient and actor-critic algorithms for MORL. For the policy gradient actor, we obtain the gradient using chain rule, and we learn different critics for each of the K objectives. Finally, we implement our algorithms on multiple environments including deep-sea treasure, and network scheduling setups to demonstrate that the proposed algorithms can optimize non-linear scalarization of multiple objectives.

KEYWORDS

Reinforcement Learning; Multi-Objective Reinforcement Learning; Actor-Critic Algorithms; Regret Analysis

ACM Reference Format:
Mridul Agarwal, Vaneet Aggarwal, and Tian Lan. 2022. Multi-Objective Reinforcement Learning with Non-Linear Scalarization. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

In many real-world problems, an agent simultaneously optimizes multiple rewards [7, 35, 40]. Further, more often than not, the objectives can be conflicting. Typical examples for such setups include, wireless sensor networks where the node may optimize between the energy consumption and freshness of the sensed parameters [8]. Another common use cases include scheduling systems which maximize the efficiency of the system but also aim to be fair to the multiple clients [12, 15, 16].

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

A fundamental method to combine the multiple objectives is using a scalarization function [26]. This allows an agent to easily compare two polices and select the one which optimizes the scalar function of the multiple objectives. For K objective values $J^1, \cdots, J^K$, a basic choice of scalarization function is $\sum_k w_k J^k$, or a linear combination of the multiple objectives. The linear combination allows the agent to use standard RL algorithms such as Q-learning [38] on the scalarized joint objective. The agent can now obtain optimal policy for any particular choice of $w_1, \cdots, w_K$. Much of the current literature focuses on finding optimal polices with linear preferences of the objectives [35, 40].

Another choice for scalarization function are non-linear monotonically increasing functions [26]. Non-linear functions have widely considered in the field of economics. Utilities of certain goods and services are often calculated using a concave function. The essential idea is they capture the natural condition of decreasing marginal utilities [36]. In this work, we propose a framework to learn policies which maximizes a concave function (not necessarily monotone) of the long-term average rewards of the multiple objectives the agent wishes to optimize.

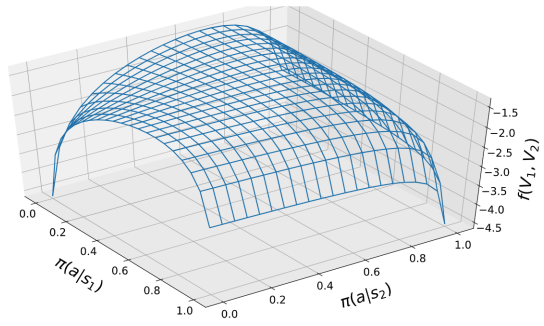


Figure 1: Proportional fairness for a 2 user, 2 state resource sharing process. The optimal strategy is not deterministic as the maximum does not lie on the boundary of the $\pi(a|s_1) - \pi(a|s_2)$ grid.

When the scalarization function is non-linear, the optimal Markov policies may not be deterministic [26]. As an example, consider an extreme case where a scheduler needs to fairly allocate a resource between two users, and the system has only one state. In this problem, we let the two objectives be the average individual long-term rewards of the two users, respectively. The two objectives are combined using a fairness objective, e.g., proportional-fair objective ($\sum_k \log J^k$, where J^k is the average long-term reward of user k). A deterministic policy will allocate the resource to only one of the users, and hence is not optimal in terms of fairness. Extension

to two state is provided in Figure 1 for a 2 user resource sharing process (Implementation details are provided in the supplementary material.The optimal policy is not deterministic as the optima does not lie on the vertices of the probability simplex.

To optimize the concave utilities of the multiple long-term average rewards, we write the long-term average rewards as a dot product of the steady state distribution of the state-action pairs and the average rewards obtained in the state-action pairs [25]. This allows to formulate a constrained convex optimization problem to solve for the steady-state distribution, and in-turn the optimal policy. To minimize the regret, the agent may try to use the optimism in the face of uncertainty [9]. However, to search for the optimistic dynamics and maximize the function requires additional computational complexity [13]. Hence, to minimize regret, we propose a model-based posterior sampling algorithm which runs in epochs. At every epoch, the algorithm samples transition probabilities, and solves the optimization equation and uses the optimal policy to interact with the environment. Further, we use a novel Bellman error based analysis to bound the regret of the algorithm instead of the standard gain-bias based analysis proposed in [9].

Using the model-based algorithm, the agent can only interact with an environment with finite state and actions. To eliminate this restriction, we consider a policy gradient algorithm which uses a simple yet elegant chain-rule to obtain true gradients of the scalarization function with respect to the policy parameters. The proposed policy gradient algorithm can be efficiently implemented using deep neural networks. It is well known that policy gradient algorithms results in high-variance [14, 28, 39] in the gradient estimates. To mitigate this issue, we derive an optimal baseline and develop an actor-critic algorithm to maximize the scalarized objective. The proposed actor-critic algorithm shares its actor architecture with the proposed policy gradient algorithm. The critic architecture learns K critic head networks to learn the value functions for the K objectives.

We summarize the contributions of this work as:

(1) A per-step reward based framework to maximize the concave utilities of multiple objective.
(2) A model-based algorithm that learns the transition probability. To reduce the computational complexity, the algorithm uses posterior sampling for MDPs with Dirichlet priors.
(3) A novel Bellman error based analysis, using which we show that the expected gap between the the overall combined objective using the model-based algorithm and the optimal combined objective reduces as $\tilde{O}(KLDS\sqrt{A/T})$.
(4) For element-wise monotone scalarization function, the optimal policy is shown to be Pareto optimal.
(5) A model-free algorithm, that uses policy gradients to find the optimal policy and can be implemented using (deep) neural networks for large state space.
(6) An actor-critic algorithm, with K critic heads, to minimize the variance of the gradients for a faster training of the actor network.

We also demonstrate the effectiveness of our proposed framework on various problems including the standard deep-sea treasure [34]. We also consider fairness maximization problems in scheduling applications [15] to demonstrate the practical use cases of the proposed algorithms. We compare the proposed algorithms with scalarized Q-learning [35], policy gradients, and actor-critic algorithm which directly attempt to maximize the scalarization function values as rewards. We show that the proposed algorithms and the gradient computations significantly outperforms the existing algorithms.

The rest of the paper is organized as follows. We first discuss the related works in detail and their differences with our work in Section 2. We formally introduce the problem formulation in Section 3. We then present the model based algorithm and the convergence guarantees in Section 4. This is followed by the model-free policy gradient algorithm in Section 5, and the actor-critic algorithm in Section 6. In Section 7, we present the evaluation results.

2 RELATED WORK

Reinforcement learning for single objective has been extensively studied in past [30]. Dynamic Programming was used in many problems by finding cost to go at each stage [4, 25]. These models optimize linear additive utility and utilize the power of Backward Induction.

Following the success of Deep Q Networks [21], many new algorithms have been developed for reinforcement learning [17, 27, 29, 37]. These papers focus on single objective control, and provide a framework for implementing scalable algorithms. Sample efficient algorithms based on rate of convergence analysis have also been studied for model based RL algorithms [3, 23], and for model free Q learning [10]. However, sample efficient algorithms use tabular implementation instead of a deep learning based implementation.

However, many practical setups require simultaneous optimization of multiple objectives. This setup is referred as Multi-Objective Reinforcement Learning (MORL). Typical MORL setups involves combining the multiple objectives using a scalarization function [26, 34, 35]. A common choice for the scalarization function is to use a weighted linear combination of the multiple objectives [26, 33, 40]. For linear combination setup, when the weights are known beforehand, the problem reduces to the standard reinforcement learning algorithm. When the weights are not known apriori, prior works attempt to learn a class of policies that work for a set of weights [2, 40]. Compared to linear preferences, we use non-linear scalarization function for our work.

For general Pareto frontiers, [24] proposed an algorithm which samples neural network parameters from distribution and updates the distribution to generate parameters corresponding to policies which achieve the Pareto Frontier. To reduce the sample complexity, they reuse the samples collected using Importance Sampling. Compared to them, we operate in the parameter space with the knowledge of the scalarization function to update the model parameter. Recently, [1] proposed a policy iteration method using which they obtain a policy that improves each objective by finding the optimal action following the current policy. Compared to the prior works, our work directly uses the scalarization function to obtain Bayesian regret bounds for a model-based tabular setup, and provide optimal state-dependent baselines for model-free policy gradient approach.

For the setup where the scalarization function is non-linear, the optimal policies may be stochastic [26] or non-stationary [7]. Hence, finding optimal policies using standard single objective reinforcement learning algorithms may become intractable. Many works cast the scalarization function as a function of average per-step rewards and steady-state. [18] studied MORL with the scalarization function of long-term average rewards and present asymptotic guarantees of their algorithm. [32] studied ergodic MDPs, or MDPs where every stationary policy results in an ergodic Markov chain over the states, to balance the state exploration.

For scalarization function as functions of long-term average rewards, [7] present a Frank-Wolfe based algorithm which uses dynamic linear preferences as the gradient of the scalarization function. They also provided non-asymptotic results for their algorithm. Recently, [41] provided an algorithm for finite horizon multi-objective RL with non-linear scalarization which uses dual updates to solve for a dynamic policy after every episode. However, their algorithm cannot be used in a train and deploy setup where an agent uses a policy trained by another agent. [5] provided an optimism based algorithm which solves for the optimal policy for a finite horizon MDP with concave utility. Their algorithm could be extended to multiple objectives, however, it is computationally complex as it involves finding the optimistic policy. In this paper, we consider stochastic policies which can be used in a train-then-test setup. To learn the stochastic policies for unknown MDPs with non-linear scalarization, we use posterior sampling to reduce the computational burden and average per-step reward criteria to solve for optimal policies. Additionally, we present the regret guarantees for the learned stochastic policies.

Prior MORL algorithm which use deep neural networks mostly focus on learning class of policies which work on linear combination scalarization functions. [2, 40] presented algorithm which learns policies for a convex converage set, or a set of policies in which all policy is optimal for some linear combination. In contrast, we study policy gradient algorithms which can work with non-linear scalarization functions. We also present the baseline which help in reducing the variance of the gradient estimates.

3 PROBLEM FORMULATION

We consider an infinite horizon discounted Markov decision process (MDP) $\mathcal{M}$, defined by the tuple $\left(\mathcal{S}, \mathcal{A}, P, K, r^1, r^2, \cdots, r^K, \gamma, \rho_0, D, f\right)$. $\mathcal{S}$ denotes a finite set of state space with $|\mathcal{S}| = S$, and $\mathcal{A}$ denotes a finite set of actions with $|\mathcal{A}| = A$. $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ denotes the probability transition distribution. We use $[K] = \{1, 2, \cdots, K\}$ to denote the set of K objectives. $r^k : \mathcal{S} \times \mathcal{A} \to [0, 1]$ denotes reward generated by objective $k \in [K]$. γ is the discount factor and $\rho_0 : \mathcal{S} \to [0, 1]$ is the distribution of initial state. $f : \mathbb{R}^K :\to \mathbb{R}$ denotes the scalarization function. D is the diameter of the MDP $\mathcal{M}$, which is the maximum expected number of steps needed to reach any state $s' \in \mathcal{S}$ from some state $s \in \mathcal{S}$. We also assume that the diameter D of the MDP, $\mathcal{M}$, is bounded and the Markov Chain induced by any stationary policy is irreducible.

We use a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, which returns the probability of selecting action $a \in \mathcal{A}$ for any given state $s \in \mathcal{S}$. The expected long term reward and expected per step reward of the objective k are given by J_π^k and λ_π^k, respectively, when the policy π is followed. Formally, J_π^k and λ_π^k are defined as

$$J_\pi^k = \mathbb{E}_{s_0, a_0, s_1, a_1, \cdots}\left[\lim_{\tau\to\infty} \sum\nolimits_{t=0}^{\tau} \gamma^t r^k(s_t, a_t)\right] \tag{1}$$

$$\lambda_\pi^k = \mathbb{E}_{s_0, a_0, s_1, a_1, \cdots}\left[\lim_{\tau\to\infty} \frac{1}{\tau}\sum_{t=0}^{\tau} r^k(s_t, a_t)\right] = \lim_{\gamma\to 1}(1-\gamma)J_\pi^k \tag{2}$$

$$s_0 \sim \rho_0(s_0),\ a_t \sim \pi(a_t|s_t),\ s_{t+1} \sim P(s_{t+1}|s_t, a_t)$$

Equation (2) follows from the Laurent series expansion of J_π^k using Corollary 8.2.4 of [25]. For brevity, in the rest of the paper, $\mathbb{E}_{s_t, a_t, s_{t+1}; t\geq 0}[\cdot]$ will be denoted as $\mathbb{E}_{\rho,\pi,P}[\cdot]$, where $s_0 \sim \rho_0(s_0),\ a_t \sim \pi(s_t|a_t),\ s_{t+1} \sim P(s_{t+1}|s_t, a_t)$.

The objectives aim to collaboratively optimize the scalarization function f, which is defined over the long-term rewards of the individual objectives. We make certain practical assumptions on this scalarization function f, which are listed as follows:

Assumption 1. *The scalarization function f is jointly concave. Hence for any arbitrary distribution $\mathcal{D}$, the following holds.*

$$f\left(\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{x}]\right) \geq \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[f(\mathbf{x})\right];\ \mathbf{x} \in \mathbb{R}^K \tag{3}$$

The objective function f represents the utility obtained from the expected per step reward of each objective. Many practically implemented fairness objectives are concave, for example cellular scheduling uses proportional fairness [15]. To model this concave utility function of the long-term average rewards, we assume the above form of Jensen's inequality. For optimizing risk, negative variance can be maximized. The negative of the variance of the K long-term average rewards will satisfy our concavity assumption and hence is a special case of our formulation.

Assumption 2. *The function f is assumed to be a L− Lipschitz function, or*

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L\left\|\mathbf{x} - \mathbf{y}\right\|_1;\ \mathbf{x}, \mathbf{y} \in \mathbb{R}^K \tag{4}$$

Lipschitz continuity is a common assumption for optimization literature [6, 11]. Additionally, in practice this assumption is validated, often by adding some regularization.

Based on these assumptions, and to keep the formulation independent of time horizon or γ, we maximize the function over expected per-step rewards of each objective. Hence, our goal is to find the optimal policy as the solution for the following optimization problem.

$$\pi^* = \arg\max_\pi f(\lambda_\pi^1, \cdots, \lambda_\pi^K) \tag{5}$$

Any online algorithm starting with no prior knowledge will require to obtain estimates of transition probabilities P and obtain rewards $r_k, \forall\ k \in [K]$ for each state action pair. Initially, when algorithm does not have good estimates of the model, it accumulates a regret for not being efficient for joint objectives as it does not know the optimal policy. We define a time dependent regret R_T to achieve an optimal solution defined as the difference between the optimal value of the function and the value of the function at time T, or

$$R_T = \left| f\left(\lambda_{\pi^*}^1, \cdots, \lambda_{\pi^*}^K\right) - f\left(\frac{1}{T}\sum_{t=0}^{T} r^1(s_t, a_t), \cdots, \frac{1}{T}\sum_{t=0}^{T} r^K(s_t, a_t)\right)\right| \tag{6}$$

We also note that the proposed framework allows for obtaining Pareto optimal policies using non-linear scalarization function in the following sub-section.

3.1 Pareto Optimality of the proposed framework

If $f(\cdot)$ is also element-wise monotone, we note that the optimal policy in (5) can be shown to be Pareto optimal. We define Pareto optimal strategy as follows.

Definition 1. *A policy π^* is said to be **Pareto optimal** if and only if there is exists no other policy π such that the average per-step reward is at least as high for all agents, and strictly higher for at least one agent. Or*

$$\forall\ k \in [K],\ \lambda^k_{\pi^*} \geq \lambda^k_{\pi}\ \textit{and}\ \exists\ k,\ \lambda^k_{\pi^*} > \lambda^k_{\pi} \tag{7}$$

The following result shows that the optimal policy satisfying Equation (5) is Pareto optimal when the function is element-wise strictly increasing.

Theorem 1. *If f is an element-wise monotonically strictly increasing function, i.e., for all $x^k > y^k$, we have*

$$f\left(\cdots, x^k, \cdots\right) > f\left(\cdots, y^k, \cdots\right), \tag{8}$$

then the solution of Equation (5)*, or the optimal policy π^* is Pareto Optimal.*

Proof. We will prove the result using contradiction. Let π^* be the solution of Equation (5) and not be Pareto optimal. Then there exists some policy π for which the following equation holds,

$$\forall\ k \in [K],\ \lambda^k_{\pi} \geq \lambda^k_{\pi^*}\ \textit{and}\ \exists\ k,\ \lambda^k_{\pi} > \lambda^k_{\pi^*} \tag{9}$$

From element-wise monotone increasing property, we obtain

$$f(\cdots, \lambda^k_{\pi}, \cdots) > f(\cdots, \lambda^k_{\pi^*}, \cdots) = \arg\max_{\pi} f(\lambda^1_{\pi}, \cdots, \lambda^K_{\pi}) \tag{10}$$

This is a contradiction. Hence, π^* is a Pareto optimal solution. □

This result shows that algorithms presented in this paper can also be used to optimally allocate resources among multiple agents using average per step allocations. We further note that the element-wise monotonically strictly increasing function assumption is only needed for the Pareto optimality of the strategy, and are not needed for the rest of the results in this paper.

After discussing the Pareto optimality, in the following section, we present a model-based algorithm to obtain this policy π^*, and regret accumulated by the algorithm.

4 MODEL-BASED ALGORITHM

Even though we work with non-linear scalarization function, we can still leverage the linear additive property of the individual long-term average reward for each objective ($\frac{1}{\tau}\sum_{t=0}^{\tau} r^k(s_t, a_t)$). Thus, the value function for individual objectives can be obtained using backward induction. For infinite horizon optimization problems (or $\tau \to \infty$), we can use steady state distribution of the state to obtain expected cumulative rewards. For all $k \in [K]$, we use

$$\lambda^k_{\pi} = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} r^k(s,a) d_{\pi}(s,a) \tag{11}$$

where $d_\pi(s,a)$ is the steady state joint distribution of the state and actions under policy π. Thus, we have the joint optimization problem in the following form

$$\max_{d_\pi(s,a)} f\Big(\sum_{s\in\mathcal{S},a\in\mathcal{A}} r^1(s,a)d_\pi(s,a), \cdots, \sum_{s\in\mathcal{S},a\in\mathcal{A}} r^K(s,a)d_\pi(s,a)\Big) \tag{12}$$

with the following set of constraints,

$$\sum_{a\in\mathcal{A}} d_\pi(s',a) = \sum_{s\in\mathcal{S},a\in\mathcal{A}} P(s'|s,a)d_\pi(s,a) \tag{13}$$

$$\sum_{s\in\mathcal{S},a\in\mathcal{A}} d_\pi(s,a) = 1, \quad d_\pi(s,a) \geq 0 \tag{14}$$

for all $s' \in \mathcal{S}$, $\forall\ s \in \mathcal{S}$, and $\forall\ a \in \mathcal{A}$. Since $f(\cdots)$ is jointly concave, arguments in Equation (12) are linear, and the constraints in Equation (13) and Equation (14) are linear, this is a convex optimization problem. Since convex optimization problems can be solved in polynomial time [6], we can use standard approaches to solve Equation (12). After solving the optimization problem, we obtain the optimal policy from the obtained steady state distribution $d^*(s,a)$ as,

$$\pi^*(a|s) = \frac{Pr(a,s)}{Pr(s)} = \frac{d^*(a,s)}{\sum_{a'\in\mathcal{A}} d^*(s,a')} \tag{15}$$

Remark 1. *Because of linear constraints if the function f is taken as* min*, the optimization problem becomes a linear program as shown in [42].*

Since we assumed that the induced Markov Chain is irreducible for all stationary policies, we assume Dirichlet distribution as prior for the state transition probability $P(s'|s,a)$. Dirichlet distribution is also used as a standard prior in literature [3, 23]. Proposition 1 formalizes the result of the existence of a steady state distribution when the transition probability is sampled from a Dirichlet distribution

Proposition 1. *For MDP $\widehat{\mathcal{M}}$ with state space $\mathcal{S}$ and action space $\mathcal{A}$, let the transition probabilities $\hat{P}$ come from Dirichlet distribution. Then, any stationary policy π for $\widehat{\mathcal{M}}$ will have a steady state distribution $\hat{d}_\pi$ given as*

$$\hat{d}_\pi(s') = \sum_{s\in\widehat{\mathcal{S}}} \hat{d}_\pi(s)\left(\sum_{a\in\widehat{\mathcal{A}}} \pi(a|s)P(s,a,s')\right) \forall s' \in \widehat{\mathcal{S}}.$$

Proof. Transition probabilities $P(s,a,\cdot)$ follow Dirichlet distribution, and hence they are strictly positive. Further, as the policy $\pi(a|s)$ is a probability distribution on actions conditioned on state, $\pi(a|s) \geq 0$, $\sum_a \pi(a|s) = 1$. So, there is a non zero transition probability to reach from state $s \in \widehat{\mathcal{S}}$ to state $s' \in \widehat{\mathcal{S}}$. Since the single step transition probability matrix is strictly positive for any policy π, a steady state distribution exists for any policy π. □

The complete model-based algorithm, named Non-Linear Scalarization-MORL-PSRL is described in Algorithm 1. The algorithm proceeds in epochs, and a new epoch is started whenever the visitation count in epoch e, $\nu_e(s,a)$, is at least the total visitations before episode e, $N_e(s,a)$, for any state action pair (Line 8). In Line 9, we sample transition probabilities $\widehat{P}$ using the updated posterior and in Line 10, we update the policy using the optimization problem specified in Equation (12).

Algorithm 1 Non-Linear Scalarization-MORL-PSRL

Input: $\mathcal{S}$, $\mathcal{A}$, $[K]$, f
Initialize $N(s,a,s') = 1\ \forall (s,a,s') \in \mathcal{S}\times\mathcal{A}\times\mathcal{S}$, $\pi(a|s) = \frac{1}{|\mathcal{A}|}\ \forall (a,s)\in\mathcal{A}\times\mathcal{S}$, $e=0$, $v_e(s,a) = N_e(s,a) = 0\ \forall (s,a)\in\mathcal{S}\times\mathcal{A}$
for time index $t = 1, 2, \cdots$ **do**
  Observe state s
  Play action $a \sim \pi(\cdot|s)$
  Observe rewards $\{r^k\}$ and next state s'
  $v_e(s,a) += 1$, $N(s,a,s') += 1$
  if $v_e(s,a) \geq \max(1, N_e(s,a))$ for any s, a **then**
    $\widehat{P}(s|a,s') \sim Dir(N(s,a,s'))\ \forall\ (s,a,s')$
    Solve steady state distribution $d(s,a)$ as the solution of the optimization problem in Equations (12-14)
    Obtain optimal policy π as

$$\pi(a|s) = \frac{d(s,a)}{\sum_{a'\in\mathcal{A}} d(s,a')}$$

    $e = e + 1$
    $v_e(s,a) = 0$, $N_e(s,a) = \sum_{e'=0}^{e} v_{e'}(s,a)\ \forall (s,a)$
  end if
end for

4.1 Regret

We now prove the regret bounds for Algorithm 1. We first give the high level ideas used in obtaining the bounds on regret. We start by dividing the regret into regret incurred in each epoch e. Then, we use the posterior sampling lemma (Lemma 1 from [23]) to obtain the equivalence between the value of the function f for the optimal policy of the true MDP $\mathcal{M}$ and the value of the function for the optimal value of the sampled MDP $\widehat{\mathcal{M}}$.

To compute the regret incurred by the optimal policy $\tilde{\pi}_e$ for the sampled MDP on the true MDP $\mathcal{M}$, we define Bellman error $B^{\pi,\tilde{P}}(s,a)$ for the infinite horizon MDPs as the difference between the cumulative expected rewards obtained for deviating from the system model with transition $\tilde{P}$ for one step by taking action a in state s and then following policy π. We have:

$$B^{\pi,\tilde{P}}(s,a) = \lim_{\gamma\to 1}\Big(Q_\gamma^{\pi,\tilde{P}}(s,a) - r(s,a) - \gamma\sum\nolimits_{s'\in\mathcal{S}} P(s'|s,a)V_\gamma^{\pi,\tilde{P}}(s,a)\Big) \tag{16}$$

We relate the Bellman error defined in Equation (16) to the gap between the expected per step reward $\tilde{\lambda}_{\tilde{\pi}_e}$ for running policy $\tilde{\pi}_e$ on sampled MDP and the expected per step reward $\lambda_{\tilde{\pi}_e}$ for running policy $\tilde{\pi}_e$ on the true MDP in the following lemma:

Lemma 1. *The difference of long-term average rewards for running the policy $\tilde{\pi}_e$ on the MDP, $\tilde{\lambda}_{\tilde{\pi}_e}$, and the average long-term average rewards for running the policy $\tilde{\pi}_e$ on the true MDP, $\lambda_{\tilde{\pi}_e}$, is the long-term average Bellman error as*

$$\tilde{\lambda}_{\tilde{\pi}_e} - \lambda_{\tilde{\pi}_e} = \sum_{s,a} d_{\tilde{\pi}_e} B^{\pi_e,\tilde{P}_e}(s,a) \tag{17}$$

where $d_{\tilde{\pi}_e}$ is the occupancy measure generated by policy $\tilde{\pi}_e$ on the true MDP.

Use the Lemma 1, and Posterior Sampling Lemma from [23], we can now bound the regret of MORL-PSRL algorithm in the form of following theorem.

Theorem 2. *The expected regret $\mathbb{E}[R_T]$ of Algorithm 1 is bounded as*

$$\mathbb{E}[R_T] \leq O\left(LDKS\sqrt{\frac{A\log T}{T}}\right) \tag{18}$$

Proof Outline. We use the Lipschitz continuity of the function to break the scalarized objective into long-term average reward regrets of individual objecectives. Using Lipschitz continuity, the total regret becomes the sum of individual regrets.

$$\mathbb{E}[R_T] = \mathbb{E}\left[\left|f\left(\cdots,\lambda^k_{\pi^*},\cdots\right) - f\left(\cdots,\frac{1}{T}\sum_{t=0}^{T} r^k(s_t,a_t),\cdots\right)\right|\right] \tag{19}$$

$$= \mathbb{E}\left[\frac{L}{T}\sum_{k=1}^{K}\left|T\lambda^k_{\pi^*} - \sum_{t=0}^{T} r^k(s_t,a_t)\right|\right] \tag{20}$$

$$\leq \frac{LK}{T}\max_{k\in[K]}\mathbb{E}\left[\left|T\lambda^k_{\pi^*} - \sum_{t=0}^{T} r^k(s_t,a_t)\right|\right]. \tag{21}$$

Following this, we break the cumulative regret into the regret incurred in each epoch e, and then use Posterior Sampling Lemma from [23] to get,

$$\mathbb{E}\left[\left|T\lambda^k_{\pi^*} - \sum_{t=0}^{T} r^k(s_t,a_t)\right|\right] = \mathbb{E}\left[\left|\sum_{e=1}^{E}\sum_{t=t_e}^{t_{e+1}-1}\left(\lambda^k_{\pi^*} - r^k(s_t,a_t)\right)\right|\right] \tag{22}$$

$$\leq \mathbb{E}\left[\left|\sum_{e=1}^{E}\sum_{t=t_e}^{t_{e+1}-1}\left(\tilde{\lambda}^k_{\tilde{\pi}_e} - r^k(s_t,a_t)\right)\right|\right]. \tag{23}$$

We now break the regret into two terms as follows:

$$\mathbb{E}\left[\left|\sum_{e=1}^{E}\sum_{t=t_e}^{t_{e+1}-1}\left(\tilde{\lambda}^k_{\tilde{\pi}_e} - r^k(s_t,a_t)\right)\right|\right] \leq \mathbb{E}\left[\left|\sum_{e=1}^{E}\sum_{t=t_e}^{t_{e+1}-1}\left(\tilde{\lambda}^k_{\tilde{\pi}_e} - \lambda^k_{\tilde{\pi}_e} + \lambda^k_{\tilde{\pi}_e} - r^k(s_t,a_t)\right)\right|\right] \tag{24}$$

The first term denotes the gap of running the optimal policy for the sampled policy on the true MDP in an epoch e. We bound this term with the Bellman error defined in Equation (16). The second term denotes the regret incurred from the deviation of the observed rewards and the expected per step rewards. The complete proof provided in the supplementary material. □

We note that for the fundamental setup of single objective with linear scalarization function ($K = 1, L = 1$), the bound becomes similar to that of UCRL2 algorithm [9].

5 MORL MODEL FREE ALGORITHM

In the previous section, we developed a model based tabular algorithm for joint function optimization. However, as the state space, action space, or number of agents increase, the tabular algorithm becomes infeasible to implement. In this section, we consider a policy gradient based algorithm which can be efficiently implemented

Algorithm 2 Non-Linear Scalarization MORL-PG

1: **Input:**$\mathcal{S}$, $\mathcal{A}$, $[K]$, T, γ, f, N, η
2: Initialize $\pi_{\theta_0}(a, s)$ with random weights θ
3: **for** $i = 0, 1, \cdots,$ $until\ convergence$ **do**
4: Collect N trajectories using policy π_{θ_i}
5: Estimate gradient using Equation (31)
6: Perform Gradient Ascent as

$$\theta_{i+i} = \theta_i + \eta \widehat{\nabla}_\theta f, \tag{25}$$

7: **end for**
8: Return π_θ

using (deep) neural networks thus alleviating the requirement of a tabular solution for large MDPs.

We note that in many practical scenario as RL environments are typically halted after a certain number of steps. Hence, for the model free policy gradient algorithm, we will use finite time horizon MDP, or $T < T$ in our MDP $\mathcal{M}$. We now describe a model free construction to obtain the optimal policy. We use a neural network parameterized by θ. The objective thus becomes to find optimal parameters θ^*, which maximizes,

$$\arg\max_\theta f\left((1-\gamma)J^1_{\pi_\theta}, \cdots, (1-\gamma)J^K_{\pi_\theta}\right). \tag{26}$$

For the model-free algorithm, we assume that the scalarization function is differentiable. In case the function is not differentiable (such as maximin fairness), sub-gradients of f can be used to optimize the objective [22]. Gradient for Equation (26) can be obtained using chain rule:

$$\nabla_\theta f = \sum\nolimits_{k\in[K]} \frac{\partial f}{\partial J^k_\pi} \nabla_\theta J^k_\pi \tag{27}$$

$$= (\nabla_{\bar{J}_\pi} f)^T (\nabla_\theta \bar{J}_\pi), \quad \bar{J}_\pi = (J^1_\pi, \cdots, J^K_\pi)^\intercal \tag{28}$$

Note that, $\bar{J}_\pi$ is the expected cumulative reward. $\bar{J}_\pi$ can be replaced with averaged cumulative rewards over N trajectories for the policy at i^{th} step, where a trajectory τ is defined as the tuple of observations, or $\tau = (s_0, a_0, r^1_0, \cdots, r^K_0, s_1, a_1, r^1_1, \cdots, r^K_1, \cdots)$. Further, $\nabla_\theta \bar{J}_\pi$ is estimated using REINFORCE algorithm proposed in [31, 39], and is given as

$$\widehat{\nabla}_\theta \bar{J}_\pi = \frac{1}{N} \sum\nolimits_{j=1}^N \sum\nolimits_{t=0}^T \nabla_\theta \log \pi_\theta(a_{t,j}|s_{t,j}) \sum\nolimits_{\tau=t}^T \gamma^\tau \bar{r}(s_{\tau,j}, a_{\tau,j}), \tag{29}$$

where $s_{\tau,j}$ and $a_{\tau,j}$ are the state and actions at time step τ of trajectory j respectively. Further, $\bar{J}_\pi$ is estimated as

$$\hat{\bar{J}} = \frac{1}{N} \sum\nolimits_{j=0}^N \sum\nolimits_{t=0}^T \gamma^t \bar{r}(s_{t,j}, a_{t,j}). \tag{30}$$

Thus, the overall estimate of $\nabla_\theta f$, $\widehat{\nabla}_\theta f$ is given as

$$\widehat{\nabla}_\theta f = \left(\nabla_{\bar{J}_\pi} f\left(\hat{\bar{J}}\right)\right)^\intercal \left(\widehat{\nabla}_\theta \bar{J}_\pi\right), \tag{31}$$

where $\widehat{\nabla}_\theta \bar{J}_\pi$ is given in (29) and $\hat{\bar{J}}$ is given in (30). On a careful inspection, we note that the gradient in Equation (31) takes the from of the Frank-Wolfe reward [7], and hence the proposed MORL-PG algorithm can be used to extend the TFW-UCRL2 algorithm [7], which uses Frank-Wolfe reward, to continuous state spaces. Based on gradient change, we present the Model Free Policy Gradient algorithm for non-linear scalarized MORL in Algorithm 2. The algorithm takes as input the parameters $\mathcal{S}, \mathcal{A}, [K], T, \gamma, f$ of MDP $\mathcal{M}$, number of sample trajectories N, and learning rate η as input. The policy neural network is initialized with weights θ randomly. In optimization step of Line 6, the weights are updated using gradient ascent.

We note that the Model free policy gradient RL algorithms are notorious for their high variance in the gradient estimate [20, 28]. In the next section, we present an optimal baseline to reduce the variance of gradient estimates.

6 MORL ACTOR CRITIC ALGORITHM

To reduce the variance of the gradient estimate used in the policy gradient algorithm, a constant value is often subtracted from the reward. In the following section, we analyze the effect of subtracting a constant value (called shift) from the gradient of the scalarization function and then find the optimal shift to reduce the variance of the gradient.

To ensure that we can indeed subtract a baseline, we begin with the Policy Gradient Theorem [31] which states

$$\nabla_\theta J^k_\pi = \mathbb{E}_\pi\left[\nabla_\theta \log \pi_\theta(a|s) Q^k_\pi(s,a)\right], \tag{32}$$

$$Q^k_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^T \gamma^t r^k(s_t, a_t) | s_0 = s, a_0 = a\right].$$

Note that we are using the gradient of the expected total rewards of individual agents to calculate the gradient of the scalarization function f. Hence, the gradients of the expected total rewards of individual agents impacts the gradient of function f via a linear function, where the weights are dependent on the function f. We first show that a state dependent shift $b(s)$ in $Q^k_\pi(s,a)$ for any agent k does not change the expected value of the gradient of the expected total rewards of individual agents. This combined with the chain rule shows that the shift in turn does not bias the gradient of the scalarization function f. Hence we can easily subtract $b(s)$ from $Q^k_\pi(s,a)$ for all k without biasing the gradient of f. The result is formalized in the following lemma.

Lemma 2. *Subtracting a constant term $b(s)$ from $Q^k_\pi(s,a), \forall k \in \{1, \cdots, K\}$ does not bias the gradient of joint objective in Equation (28).*

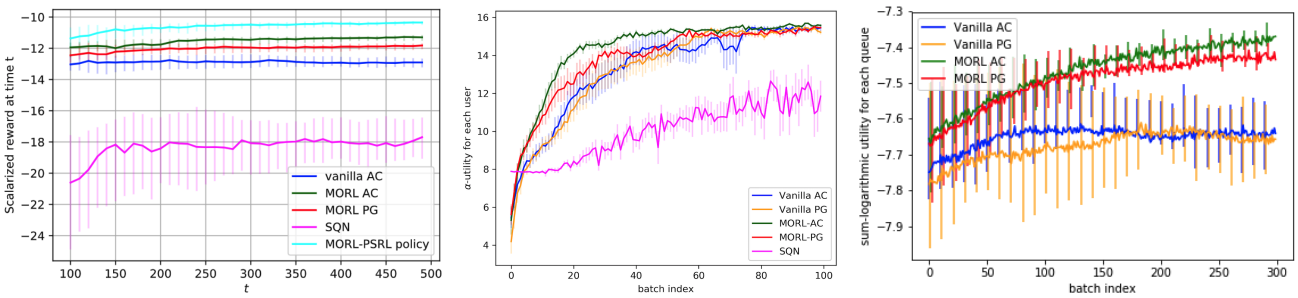
Proof. Using the policy gradient theorem in Equation (33), we have,

$$\nabla_\theta f = \sum_{k\in[K]} \frac{\partial f}{\partial J^k_\pi} \nabla_\theta J^k_\pi \tag{33}$$

$$= \sum_{k\in[K]} \frac{\partial f}{\partial J^k_\pi} \mathbb{E}_\pi\left[\nabla_\theta \log \pi_\theta(a|s) Q^k_\pi(s,a)\right] \tag{34}$$

To show that that shift does not bias the gradient, we want to show that,

$$\sum_{k=1}^K \frac{\partial f}{\partial J^k_\pi} \mathbb{E}_\pi\left[\nabla_\theta \log \pi_\theta(a|s)\left(Q^k_\pi(s,a) - b(s)\right)\right] = \nabla_\theta f$$



(a) Wireless Scheduling (b) Deep Sea Treasure (c) Queuing System

Figure 2: Learning curves for the three environments considered for the proposed algorithm against standard Actor Critic, Policy Gradient, and SQN algorithms. The proposed algorithm learns policies which maximizes the scalarized rewards in all three setups. (a) Wireless Scheduling: For this setup we plot the performance of the learned policies using multiple algorithms. (b) For the episodic Deep Sea Treasure problem, we compare the MORL-PG algorithm with Scalarized-DQN (SDQN) [35]. (c) For the queuing system we compare MORL-AC algorithm with MORL-AC, and their vanilla implementations.

Hence, it would suffice to show that $\mathbb{E}_\pi\left[\nabla_\theta \log \pi_\theta(a|s) b(s)\right] = 0$.

$$\begin{aligned}
&\mathbb{E}_\pi\left[\nabla_\theta \log \pi_\theta(a|s) b(s)\right] \\
&= \mathbb{E}_s\left[\mathbb{E}_{a\sim\pi(\cdot|s)}\left[\nabla_\theta \log \pi_\theta(a|s) b(s)\right] \middle| s\right] && (35)\\
&= \mathbb{E}_s\left[b(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) \middle| s\right] && (36)\\
&= \mathbb{E}_s\left[b(s) \sum\nolimits_{a\in\mathcal{A}} \nabla_\theta \pi_\theta(a|s) \middle| s\right] && (37)\\
&= \mathbb{E}_s\left[b(s) \nabla_\theta \sum\nolimits_{a\in\mathcal{A}} \pi_\theta(a|s) \middle| s\right] && (38)\\
&= \mathbb{E}_s\left[b(s) \nabla_\theta 1 \middle| s\right] = 0 && (39)
\end{aligned}$$

□

We can now attempt to find an optimal value of the shift $b(s)$ to reduce the variance of the policy gradient. The result is stated in the following theorem, with proof in the supplementary material.

Theorem 3. *The state dependent shift $b(s)$ that minimizes variance of $\nabla_\theta f$ is given as:*

$$b^*(s) = \frac{\mathbb{E}\left[G(s,a)\left(\sum_{k=1}^K \frac{\partial f}{\partial J_\pi^k} Q_\pi^k(s,a)\right)\right]}{\mathbb{E}\left[G(s,a)\left(\sum_{k=1}^K \frac{\partial f}{\partial J_\pi^k}\right)\right]} \tag{40}$$

where $G = (\nabla_\theta \log \pi(a|s))^T (\nabla_\theta \log \pi(a|s))$.

Note that the optimal shift $b^*(s)$ is similar to a weighted average of the value functions of agents in state s. In practice, it is convenient to exclude G in optimal baseline [19], and hence, we use a modified shift $\tilde{b}^*(s)$ as

$$\tilde{b}^*(s) = \frac{\sum_{k=1}^K \frac{\partial f}{\partial J_\pi^k} \mathbb{E}_\pi[Q_\pi^k(s,a)]}{\sum_{k=1}^K \frac{\partial f}{\partial J_\pi^k}} = \frac{\sum_{k=1}^K \frac{\partial f}{\partial J_\pi^k} V_\pi^k(s)}{\sum_{k=1}^K \frac{\partial f}{\partial J_\pi^k}} \tag{41}$$

Using this idea, we can now construct an actor-critic-based algorithm where, along with the actor network, we train a K head critic network V_ϕ that approximates value functions for each of the K objectives. We now describe the construction of the actor network with weights θ which will be followed by the description of the critic network with weights θ.

5.1 Actor Network

The network is almost identical to Section 5 with a baseline subtracted from the the gradient. From Equation (41), the optimal baseline is a weighted sum of the value functions of the individual objectives. We use the objective function f to determine the baseline $\hat{b}^*(s)$ as:

$$\widehat{b}^*(s_{t,j}) = \left(\sum_{k=1}^K \frac{\partial f}{\partial J_\pi^k} V_\phi^k(s_{t,j})\right)^{-1} \left(\sum_{k=1}^K \frac{\partial f}{\partial J_\pi^k}\right) \tag{42}$$

Using the baseline calculated in Equation (42) and the gradient estimate $\widehat{\nabla}_\theta f$ can be obtained as

$$\begin{aligned}
\widehat{\nabla}_\theta f &= \frac{1}{N} \sum_{j=1}^N \sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_{t,j}|s_{t,j}) \times \zeta_t && (43)\\
\zeta_t &= \left(\sum_{\tau=t}^T \gamma^\tau r^k(s_{t',j}, a_{t',j}) - \widehat{b}^*(s_{t,j})\right)
\end{aligned}$$

5.2 Critic Network

Note that, the baseline which reduces the variance of the policy gradient depends on the value function of the individual objectives. Hence we construct a critic network which learns the value function of the individual objectives. Note that to learn the critic network, we do not require the knowledge of the function.

For learning rate η_c, we can learn the critic network using gradient descent as:

$$\begin{aligned}
\phi_{i+1} &= \phi_i - \eta_c \left(\frac{1}{N} \sum_{j=0}^N \sum_{t=0}^T \left(\gamma^t \bar{r}(s_{t,j}, a_{t,j}) - V_\phi(s_{t,j})\right)\right) \\
&\text{where } \bar{r}(s_{t,j}, a_{t,j}) = (r^1(s_{t,j}, a_{t,j}), \cdots, r^K(s_{t,j}, a_{t,j}))^T
\end{aligned} \tag{44}$$

5.3 Proposed MORL Actor Critic Algorithm

The proposed actor-critic algorithm is described in Algorithm 3. The algorithm takes as input the same parameters as Algorithm 2 but with two learning rates, η_a for the actor network and η_c for the critic network. The actor neural network is initialized with random weights θ and the critic neural network is initialized with random weights ϕ. It then collects N sample trajectories using the policy

Algorithm 3 Non Linear Scalarized MORL-AC

1: **Input:** $\mathcal{S}$, $\mathcal{A}$, $[K]$, T, γ, f, N, η_a, η_c
2: Initialize $\pi_\theta(a, s)$, $V_\phi(s)$ ▷ Initialize the neural network with random weights θ for actor and ϕ for Value Estimator
3: **for** $i = 0, 1, \cdots,$ $until\ convergence$ **do**
4: Collect N trajectories using policy π_θ
5: Estimate Actor gradient using Equation (43)
6: Update Actor and Critic Networks using Equations (25) and (44)
7: **end for**
8: Return π_θ

with current weights in Line 4. Using the sampled trajectories, it obtains the gradients using the shift with updated critic network using Equation (43), and updates the actor network . After collecting the trajectories, it updates the critic network using Equation (44).

7 EVALUATIONS

We evaluate the proposed MORL algorithms on different setups. We include standard MORL setup such as Deep Sea Treasure [34], and many practical setups from wireless scheduling, and a queuing system. We use different concave utilities to demonstrate the effectiveness of the algorithm proposed on a large range of non-linear scalarization functions. We compare all the algorithm with Scalarized Q-learning [35], and a vanilla policy gradient, and a vanilla actor-critic algorithm which takes the scalarized rewards directly as rewards. Implementation details for all the algorithms is provided in the supplementary material. We begin with introducing the environments and the scalarization functions as follows:

(1.) A 2 state wireless scheduling system system with 4 users. This simple setup consists of 16 states. For this setup, we use an α-concave utility (for $\alpha = 2$) of the data transmitted to each user which is defined as:

$$f(\lambda_\pi^1, \lambda_\pi^2) = \sum_{k=1}^{4} \frac{\left(\lambda_\pi^k\right)^{1-\alpha}}{1-\alpha} \tag{45}$$

(2.) A 10×5 grid Deep Sea Treasure, which is a classic MORL benchmark. This environment is episodic where the episode ends if the treasure is found or if the maximum time allowed is consumed. The task requires maximizing the treasure R and minimizing the time t required to reach the treasure. For this task, we use the following reward scalarization.

$$f(R, t) = \sqrt{R} + \sqrt{50 - t} \tag{46}$$

(3.) A 4-queue system with heterogeneous arrival rates, with a proportional-fairness utility function of the average number of packets routed from each queue. The non-linear scalarization function becomes:

$$f((1-\gamma)J_\pi^1, \cdots, (1-\gamma)J_\pi^4) = \sum_{k=1}^{4} \log\left((1-\gamma)J_\pi^k\right) \tag{47}$$

We compare the performance of the proposed algorithms with other algorithms in Figure 2. In each sub-figure, we plot the mean and standard deviation error bars calculated using 10 independent iterations. For the wireless scheduling example, we note that the performance of Scalarized Q-learning [35] is the worst. This is because of the use of deterministic policies. The MORL-AC and MORL-PG algorithms outperforms the vanilla actor critic algorithm because of the use of the scaled gradients and giving different weights to the objective which results in a higher scalarized value. The convergence plots for the policy gradient and actor-critic algorithms are presented in the supplementary material. The performance of the MORL-PSRL is the best among all the algorithms. This is expected as the model based algorithm will converge to the optimal policy while model free algorithm can get stuck in local optimas because of parameterization.

For the episodic Deep Sea Treasure, we found that the Scalarized Q-learning algorithm is able to learn good policies which maximize the scalarized rewards because there is a unique optimal treasure which maximizes Equation 46, and the optimal policy becomes deterministic. However, the learning rate is too slow compared to the policy gradients and actor-critic algorithms. Further because of this, performance of the proposed algorithms is marginally better because standard policy gradients.

For the Queuing system, we first note that the scalarized Q-learning algorithm performs significantly worse than the standard algorithms which are able to learn stochastic policies and hence we exclude the plot for clarity of Figure 2(c). We note that the standard policy gradient and actor critic algorithms which take scalarized objectives as rewards are able to learn some policy which tries to maximize the sclarized objective. Both MORL-AC algorithm and MORL-PG algorithm outperform the standard policy gradient and actor critic algorithm. This is because the gradient dynamically weighs the objective which pushes the network to a higher scalarized reward value. Additionally, the MORL-AC algorithm converges faster compared to the MORL-PG algorithm because of reduced variance of gradient estimates.

8 CONCLUSION

This paper, considers the problem of Multi-Objective Reinforcement Learning (MORL) with non-linear scalarization using concave utilities. To optimize the concave utility, we proposed an average per step reward based formulation using long-term rewards of each agent. Under certain assumptions, the optimal policy is also Pareto Optimal. We proposed a model based algorithm that progresses in epochs and samples transition probabilities using posterior sampling for known priors. Following optimal policy for the sampled MDP at every epoch, using a novel Bellman error based analysis, we show that the proposed algorithm converges towards the optimal L-Lipschitz scalarized objective as $\tilde{O}(LKDS\sqrt{A/T})$. We also proposed a model free algorithm which can be efficiently implemented using neural networks. We find the optimal baseline to reduce the variance of the policy gradient algorithm and propose an actor-critic based algorithm. Finally, we evaluate the proposed algorithms on various scheduling problems, where we show that the proposed algorithms work well with non-linear scalarizations and outperforms existing algorithms.

9 ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation under Grant CNS-1618335 and in part by CISCO Systems, Inc.

REFERENCES

[1] Abbas Abdolmaleki, Sandy Huang, Leonard Hasenclever, Michael Neunert, Martina Zambelli, Murilo Martins, Francis Song, Nicolas Heess, Raia Hadsell, and Martin Riedmiller. 2020. A distributional view on multi objective policy optimization. In *ICML 2020: 37th International Conference on Machine Learning*, Vol. 1. 11–22.

[2] 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.

[3] Shipra Agrawal and Randy Jia. 2017. Optimistic posterior sampling for reinforcement learning: worst-case regret bounds. In *Advances in Neural Information Processing Systems*. 1184–1194.

[4] Dimitri P Bertsekas. 1995. *Dynamic programming and optimal control*. Vol. 1. Athena scientific Belmont, MA.

[5] KiantÃĪ Brantley, Miroslav DudÃŋk, Thodoris Lykouris, Sobhan Miryoosefi, Max Simchowitz, Aleksandrs Slivkins, and Wen Sun. 2020. Constrained episodic reinforcement learning in concave-convex and knapsack settings. In *Advances in Neural Information Processing Systems*, Vol. 33. 16315–16326.

[6] Sébastien Bubeck et al. 2015. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning* 8, 3-4 (2015), 231–357.

[7] Wang Chi Cheung. 2019. Regret minimization for reinforcement learning with vectorial feedback and complex objectives. *Advances in Neural Information Processing Systems* 32 (2019), 726–736.

[8] Jean-Marie Gorce, Ruifeng Zhang, Katia Jaffres-Runser, and Claire Goursaud. 2010. Energy, latency and capacity trade-offs in wireless multi-hop networks. In *21st Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE, 2757–2762.

[9] Thomas Jaksch, Ronald Ortner, and Peter Auer. 2010. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research* 11, Apr (2010), 1563–1600.

[10] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. 2018. Is q-learning provably efficient?. In *Advances in Neural Information Processing Systems*. 4863–4873.

[11] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. 2017. How to escape saddle points efficiently. In *International Conference on Machine Learning*. PMLR, 1724–1732.

[12] Carlee Joe-Wong, Soumya Sen, Tian Lan, and Mung Chiang. 2013. Multiresource allocation: fairness-efficiency tradeoffs in a unifying framework. *IEEE/ACM Trans. Netw.* (2013), 1206–1214.

[13] Krishna C Kalagarla, Rahul Jain, and Pierluigi Nuzzo. 2020. A Sample-Efficient Algorithm for Episodic Finite-Horizon MDP with Constraints. *arXiv preprint arXiv:2009.11348* (2020).

[14] Vijay R Konda and John N Tsitsiklis. 2000. Actor-critic algorithms. In *Advances in neural information processing systems*. Citeseer, 1008–1014.

[15] Raymond Kwan, Cyril Leung, and Jie Zhang. 2009. Proportional fair multiuser scheduling in LTE. *IEEE Signal Processing Letters* 16, 6 (2009), 461–464.

[16] Tian Lan, David Kao, Mung Chiang, and Ashutosh Sabharwal. 2010. *An axiomatic theory of fairness in network resource allocation*. IEEE.

[17] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[18] Shie Mannor and Nahum Shimkin. 2004. A geometric approach to multi-criterion reinforcement learning. *The Journal of Machine Learning Research* 5 (2004), 325–360.

[19] Hongzi Mao, Shaileshh Bojja Venkatakrishnan, Malte Schwarzkopf, and Mohammad Alizadeh. 2018. Variance Reduction for Reinforcement Learning in Input-Driven Environments. In *International Conference on Learning Representations*.

[20] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. 1928–1937.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[22] Yurii Nesterov. 2018. *Lectures on convex optimization*. Vol. 137. Springer.

[23] Ian Osband, Daniel Russo, and Benjamin Van Roy. 2013. (More) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*. 3003–3011.

[24] Simone Parisi, Matteo Pirotta, and Jan Peters. 2017. Manifold-based multi-objective policy search with sample reuse. *Neurocomputing* 263 (2017), 3–14.

[25] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (1st ed.). John Wiley & Sons, Inc., New York, NY, USA.

[26] Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. 2013. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research* 48 (2013), 67–113.

[27] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. 1889–1897.

[28] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).

[29] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[30] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[31] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*. 1057–1063.

[32] Jean Tarbouriech and Alessandro Lazaric. 2019. Active exploration in markov decision processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 974–982.

[33] Gerald Tesauro, Rajarshi Das, Hoi Chan, Jeffrey O Kephart, David Levine, Freeman L Rawson III, and Charles Lefurgy. 2007. Managing Power Consumption and Performance of Computing Systems Using Reinforcement Learning.. In *NIPS*, Vol. 7. Citeseer, 1–8.

[34] Peter Vamplew, Richard Dazeley, Adam Berry, Rustam Issabekov, and Evan Dekker. 2011. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine learning* 84, 1 (2011), 51–80.

[35] Kristof Van Moffaert and Ann Nowé. 2014. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research* 15, 1 (2014), 3483–3512.

[36] Vijay V Vazirani and Mihalis Yannakakis. 2011. Market equilibrium under separable, piecewise-linear, concave utilities. *Journal of the ACM (JACM)* 58, 3 (2011), 1–25.

[37] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. 2015. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581* (2015).

[38] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.

[39] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

[40] Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. 2019. A Generalized Algorithm for Multi-Objective Reinforcement Learning and Policy Adaptation. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/4a46fbfca3f1465a27b210f4bdfe6ab3-Paper.pdf

[41] Tiancheng Yu, Yi Tian, Jingzhao Zhang, and Suvrit Sra. 2021. Provably Efficient Algorithms for Multi-Objective Competitive RL. In *ICML 2021: 38th International Conference on Machine Learning*. 12167–12176.

[42] Chongjie Zhang and Julie A Shah. 2014. Fairness in multi-agent sequential decision-making. In *Advances in Neural Information Processing Systems*. 2636–2644.