REWARD DIMENSION REDUCTION FOR SCALABLE MULTI-OBJECTIVE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we introduce a simple yet effective reward dimension reduction method to tackle the scalability challenges of multi-objective reinforcement learning algorithms. While most existing approaches focus on optimizing two to four objectives, their abilities to scale to environments with more objectives remain uncertain. Our method uses a dimension reduction approach to enhance learning efficiency and policy performance in multi-objective settings. While most traditional dimension reduction methods are designed for static datasets, our approach is tailored for online learning and preserves Pareto-optimality after transformation. We propose a new training and evaluation framework for reward dimension reduction in multi-objective reinforcement learning and demonstrate the superiority of our method in an environment with sixteen objectives, significantly outperforming existing online dimension reduction methods.

1 INTRODUCTION

Reinforcement Learning (RL) is a powerful machine learning paradigm focused on training agents to make sequential decisions by interacting with their environment. Through trial and error, RL algorithms allow agents to iteratively improve their decision-making policies, with the ultimate goal of maximizing cumulative rewards. In recent years, the field of Multi-Objective Reinforcement Learning (MORL) has gained considerable attention due to its relevance in solving real-world control problems involving multiple, often conflicting, objectives. These problems span across domains such as power system management, autonomous vehicle control, and logistics optimization, where balancing trade-offs among competing objectives is crucial (Roijers et al., 2013).

MORL extends the traditional RL framework by enabling agents to handle multiple objectives simultaneously. This requires methods capable of identifying and managing trade-offs among these objectives. MORL specifically focuses on learning a set of policies that approximate the Pareto frontier, representing solutions where no objective can be improved without compromising others. Most of the current approaches scalarize vector rewards into scalar objectives to generate a diverse set of policies (Abels et al., 2019; Yang et al., 2019; Xu et al., 2020; Basaklar et al., 2023; Lu et al., 2023), thereby avoiding the need for retraining during the test phase.

Although these methods have proven effective in standard MORL benchmarks (Felten et al., 2023), most benchmarks involve only two to four objectives, leaving open the question of whether existing MORL algorithms can scale effectively to environments with more objectives (Hayes et al., 2022). Indeed, various practical applications demand optimizing many objectives simultaneously (Li et al., 2015). For example, Fleming et al. (2005) introduced an example of optimizing a complex jet engine control system that requires balancing eight physical objectives. In military contexts, a commander should manage dozens of objectives that directly influence decision-making (Dagistanli & Üstün, 2023), including the positions of allies and enemies, casualty rates, combat capabilities of allies and enemies, and time estimations for achieving strategic goals. When planning for multiple potential battle scenarios, exploring such high-dimensional objective space in its raw form is inefficient and very challenging due to the complexity of the original space.

An advantageous feature of many real-world MORL applications is that objectives often exhibit correlations, leading to inherent conflicts or trade-offs. For example, an autonomous vehicle must balance safety and speed, where optimizing one can compromise the other. Similarly, traffic light control must manage multiple interrelated objectives to ensure smooth traffic flow (Hayes et al., 2022). These correlations suggest that reducing the dimensionality of the reward space while preserving its essential features could be a viable strategy to make current MORL algorithms scalable to many objective spaces.

Dimension reduction techniques (Roweis & Saul, 2000; Tenenbaum et al., 2000; Zass & Shashua, 2006; Lee et al., 2007; Cardot & Degras, 2018; McInnes & Healy, 2018; Bank et al., 2020), widely used in other machine learning domains, capture the most significant features of high-dimensional data while filtering out irrelevant noise. However, typical approaches operate on static datasets, whereas RL necessitates continuous data collection during online training. This introduces a unique challenge: applying dimension reduction in MORL while retaining the essential structure of the original reward space. To our knowledge, few studies have addressed this challenge within the MORL context.

In this paper, we address these challenges by introducing a simple yet effective reward dimension reduction method that scales MORL algorithms to higher-dimensional reward spaces. We propose a new training and evaluation framework tailored for the online reward dimension reduction setting. Our approach ensures that Pareto-optimality is preserved after transformation, allowing the agent to learn and execute policies that remain effective in the original reward space.

Our contributions are as follows:

- We propose a new training and evaluation framework for online reward dimension reduction in MORL.
- We derive conditions and introduce learning techniques to ensure that online training and Pareto-optimality are maintained, providing a stable and efficient approach for scalable MORL.
- We conduct experiments with sixteen objectives, demonstrating superior performance compared to existing online dimension reduction methods.

2 BACKGROUND

A multi-objective Markov decision process (MOMDP) is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, P, \mu_0, r, \gamma \rangle$. Here, $\mathcal{S}$ represents the set of states, $\mathcal{A}$ the set of actions, P the state transition probabilities, μ_0 the initial state distribution, r the reward function, and $\gamma \in [0, 1)$ the discount factor. Unlike the traditional single-objective MDP, the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^K$ in a MOMDP is vector-valued, where $K \geq 2$ is the number of objectives. This vector-valued nature of the reward function allows the agent to receive multiple rewards for each state-action pair, each corresponding to a different objective.

In the context of MORL, the performance of a policy π is evaluated by its expected cumulative reward, denoted as $J(\pi) = (J_1(\pi), \cdots, J_K(\pi)) := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \in \mathbb{R}^K$. To compare vector-valued rewards, we use the notion of Pareto-dominance (Roijers et al., 2013), denoted $>_P$. For two vector returns, $J(\pi)$ and $J(\pi')$, we have:

$$J(\pi') >_P J(\pi) \iff (\forall i \in \{1, \ldots, K\}, J_i(\pi') \geq J_i(\pi)) \text{ and } (\exists j \in \{1, \ldots, K\}, J_j(\pi') > J_j(\pi)). \tag{1}$$

This means that $J(\pi')$ Pareto-dominates $J(\pi)$ if it is at least as good as $J(\pi)$ in all objectives and strictly better in at least one.

The goal of MORL is to identify a policy π whose $J(\pi)$ lies on the Pareto frontier (or boundary) $\mathcal{F}$ of all achievable return tuples $\mathcal{J} = \{(J_1(\pi), \cdots, J_K(\pi)) \mid \pi \in \Pi\}$, where Π denotes the set of all possible policies. The formal definition of the Pareto frontier [1] is as follows (Roijers et al., 2013; Yang et al., 2019):

$$\mathcal{F} = \{J(\pi) \mid \nexists \pi' \text{ s.t. } J(\pi') >_P J(\pi)\}. \tag{2}$$

---

[1] Strictly speaking, the Pareto frontier can also be defined as the set of non-dominated *policies*, $\{\pi \in \Pi \mid \nexists \pi' \text{ s.t. } J(\pi') >_P J(\pi)\}$ (Hayes et al., 2022), rather than the set of non-dominated *vector returns* as shown in equation 2 . In this case, multiple policies may achieve the same vector return (Hayes et al., 2022). To avoid this redundancy, in this paper, we define the Pareto frontier and the convex coverage set as presented in equation 2 and equation 3, respectively.

In other words, no single policy achieving $\mathcal{F}$ can improve one objective without sacrificing at least one other objective. Finding a policy achieving the Pareto frontier ensures an optimal balance among the competing objectives with the best possible trade-offs.

Researchers are also interested in obtaining policies that cover the *convex coverage set* (CCS) of a given MOMDP defined as follows (Yang et al., 2019):

$$CCS = \{J(\pi) \mid \exists \omega \in \Delta^K \text{ s.t. } \omega^\top J(\pi) \geq \omega^\top J(\pi'),\ \forall \pi' \in \Pi \text{ with } J(\pi') \in \mathcal{F}\} \tag{3}$$

where Δ^K is the $(K-1)$-simplex and $\omega \in \Delta^K$ represents a preference vector that specifies the relative importance of each objective (i.e., $\sum_{k=1}^K \omega_k = 1, \omega_k \geq 0, \forall k$).

Figure 1 illustrates the relationship between Pareto frontier and CCS. In Figure 1, we assume that the achievable points $\{A, B, C, D, E\}$ form the Pareto frontier. Then, for the preference vector ω in Figure 1, the inner product between ω and the return vector at the point C is smaller than the inner product between ω and the return vector at the point D. In fact, the inner product between ω and the return vector at the point C is smaller than that between ω and any other point in the Pareto frontier. Hence, the point C is not included in the CCS. Note that the CCS represents the set of achievable returns that are optimal for some linear combination of objectives, and it is a subset of the Pareto frontier $\mathcal{F}$ by definition. Since the weighted sum is widely used in real-world applications to express specific preferences over multiple objectives (Hayes et al., 2022), CCS is a proper refinement of the Pareto frontier.

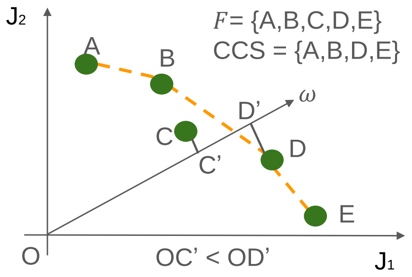

Figure 1: Comparison of the Pareto frontier $\mathcal{F}$ and the CCS for $K = 2$, where C' and D' represent the projections of points C and D onto the preference vector ω, respectively. Yellow dashed line represents the outer convex boundary of $\mathcal{F}$.

In the context of *multi-policy* MORL (Roijers et al., 2013), the goal is to find multiple policies that cover (an approximation of) either the Pareto frontier or the CCS so that during test phases, we can perform well across various scenarios without having to retrain from scratch. Specifically, we aim to achieve Pareto-optimal points that *maximize the hypervolume while minimizing the sparsity* (Hayes et al., 2022).

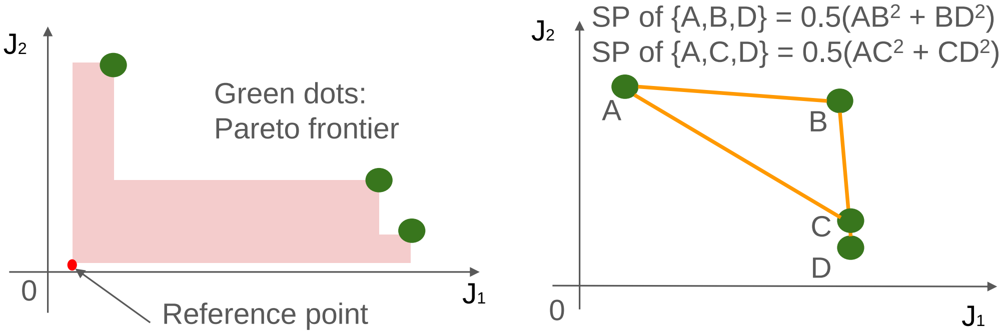
J₂
Green dots:
Pareto frontier
0
Reference point
J₁
J₂
SP of {A,B,D} = 0.5(AB² + BD²)
SP of {A,C,D} = 0.5(AC² + CD²)
A
B
C
D
0
J₁

Figure 2: Evaluation metrics in multi-policy MORL: hypervolume and sparsity. (Left) Hypervolume is represented by the pink area in the figure. (Right) The sparsity of the solution set $\{A, B, D\}$ is lower than that of $\{A, C, D\}$ when points C and D are close, indicating that $\{A, B, D\}$ offers a more diverse set of solutions than $\{A, C, D\}$.

As seen in the left figure of Figure 2, the hypervolume measures the volume in the objective space dominated by the set of current Pareto frontier points and bounded by a reference point. In the figure, the hypervolume corresponds to the area of the pink region. This metric provides a scalar value quantifying how well the policies cover the objective space. Formally, let $X = \{x_1, \cdots, x_N\} \subset \mathbb{R}^K$ be a set of N Pareto frontier points and $x_0 \in \mathbb{R}^K$ be a reference point, where $x_i = (x_{i1}, \ldots, x_{iK}),\ i = 0, \ldots, N$. Then, the hypervolume metric $HV(X, x_0)$ is defined by the K-dimensional volume of the union of hybercubes $\bigcup_{i=1}^{N} \mathcal{C}_{k=1}^{K}[x_{0k}, x_{ik}]$, where $\mathcal{C}_{k=1}^{K}[x_{0k}, x_{ik}]$ is the hypercube of which side at the k-th dimension is given by the line segment $[x_{0k}, x_{ik}]$.

Another metric is sparsity, which assesses the distribution of policies within the objective space. As seen in the right figure of Figure 2, a set of Pareto frontier points with low sparsity ensures that the solutions are well-distributed, offering a diverse range of trade-offs among the objectives. If there is a set of N Pareto frontier points $X = \{x_1, \cdots, x_N\} \subset \mathbb{R}^K$ with $x_i = (x_{i1}, \ldots, x_{iK})$ $(i = 1, \ldots, N)$, sparsity is defined as:

$$SP := \frac{1}{N-1} \sum_{k=1}^{K} \sum_{i=1}^{N-1} (S_k[i] - S_k[i+1])^2 \tag{4}$$

where $S_k = \text{Sort}\{x_{ik}, 1 \leq i \leq N\}$ in descending order in the k-th objective, $1 \leq k \leq K$. Given a dimension k and its two endpoints, $S_k[1]$ and $S_k[N]$, the Cauchy–Schwarz inequality implies that $\sum_{i=1}^{N-1}(S_k[i] - S_k[i+1])^2$ is minimized when the differences $S_k[i] - S_k[i+1]$ are constant for all $1 \leq i \leq N-1$. Therefore, sparsity acts as an indicator of how well-distributed a set of return vectors is. Reducing sparsity while maintaining a high hypervolume helps avoid situations where only a few objectives perform well. Therefore, considering both low sparsity and high hypervolume offers a more comprehensive evaluation criterion than relying solely on hypervolume.

3 RELATED WORKS

There are mainly two branches in MORL. The first branch is single-policy MORL, where the goal is to obtain an optimal policy $\pi^* = \arg\max_\pi h(J(\pi))$ where $h : \mathbb{R}^K \to \mathbb{R}$ is a *fixed* non-decreasing utility function, mostly for non-linear one (Siddique et al., 2020; Park et al., 2024). The other branch is the multi-policy MORL, where we aim to acquire multiple policies that cover an approximation of the Pareto frontier or CCS. Beyond several classical methods such as iterative single-policy approaches (Roijers et al., 2014), current approaches in the multi-policy MORL either train a set of multiple policies (Xu et al., 2020) or train a single network to cover multiple policies (Abels et al., 2019; Yang et al., 2019; Basaklar et al., 2023; Lu et al., 2023). For completeness, these approaches should be followed by a preference elicitation method for the test phase given that additional interactive approaches are allowed (Hayes et al., 2022) (e.g., Zintgraf et al. (2018) inferred unknown user preference using queries of pairwise comparison on the Pareto frontier). Nonetheless, the area of elicitation has received far less attention than the learning methods themselves. Researchers usually focus solely on the learning algorithms during the training phase assuming that test preferences will be explicitly given.

Xu et al. (2020) trains a set of multiple policies in parallel using the concept of evolutionary learning, and the best policy in the policy set is used for evaluation during the test phase. Other works construct a single policy network parameterized by $\omega \in \Delta^K$ to cover the CCS, which is easier than direct parameterization over the set of non-decreasing functions to cover the Pareto frontier. Abels et al. (2019) and Yang et al. (2019) constructed single-policy networks to exploit the advantages of CCS. Specifically, Yang et al. (2019) defined the optimal multi-objective action-value function for all $\omega \in \Delta^K$: $Q^*(s, a, \omega) = \mathbb{E}_{P,\pi^*(\cdot|\cdot;\omega)|s_0=s,a_0=a}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \in \mathbb{R}^K$, where the optimal policy π^* is given by $\pi^*(\cdot|\cdot;\omega) = \arg\sup_\pi \omega^\top \mathbb{E}_{P,\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$. Based on a new definition of the multi-objective optimality operator, the authors proposed an algorithm for training a neural network $Q_\theta(s, a, \omega)$ to approximate $Q^*(s, a, \omega)$. Basaklar et al. (2023) modified the multi-objective optimality operator to match each direction of the learned action-value function and preference vector, and Lu et al. (2023) tackled a learning stability issue of multi-policy MORL by providing theoretical analysis on linear scalarization.

While these methods have demonstrated promising performance in MORL benchmarks with two to four objectives, it remains an open question whether current algorithms can effectively scale to environments with more objectives (Hayes et al., 2022). In this paper, we address this scalability issue by proposing a reward dimension reduction technique with a suitable training and evaluation framework. Our approach is motivated by the observation that objectives are correlated in many real-world cases. While a variety of dimension reduction techniques exist in machine learning (Roweis & Saul, 2000; Tenenbaum et al., 2000; Lee et al., 2007; McInnes & Healy, 2018), most are designed for static (batch-based) datasets. Only a few methods are suitable for online settings, and in some cases, no online version exists at all (McInnes & Healy, 2018). Developing online variants of batch-based dimension reduction techniques is itself an active area of research. Currently, incremental principal component analysis (PCA) and online autoencoders are commonly used for online dimension reduction (Cardot & Degras, 2018; Bank et al., 2020). However, we will demonstrate that they fail to preserve Pareto-optimality after transformation in the context of multi-policy MORL.

To our knowledge, few studies have explored reward dimension reduction in MORL. For instance, Giuliani et al. (2014) applied non-negative principal component analysis (NPCA) to a fixed set of return vectors collected from several pre-defined scenarios, identifying the principal components. However, they did not perform any further online interactions, but multi-policy MORL algorithms require online learning. In this paper, we propose a simple yet stable method for online dimension reduction that preserves Pareto-optimality after transformation, as described in the following section.

4 METHOD

4.1 TRAINING AND EVALUATION FRAMEWORK

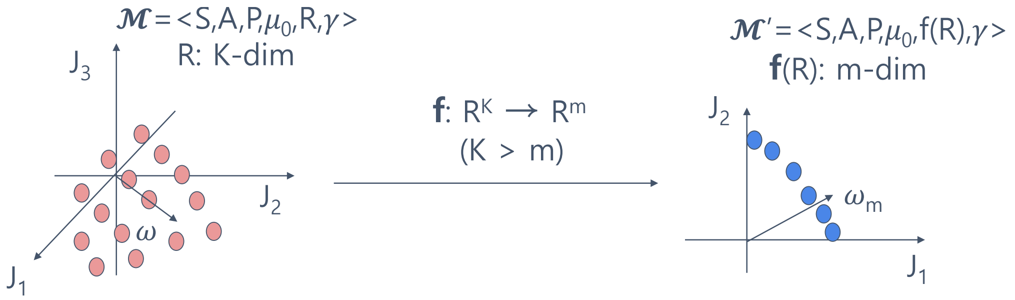
$\mathcal{M}$=<S,A,P,μ_0,R,γ>
R: K-dim
J_3
J_2
J_1
ω
f: $R^K \rightarrow R^m$
(K > m)
$\mathcal{M}'$=<S,A,P,μ_0,f(R),γ>
f(R): m-dim
J_2
ω_m
J_1

Figure 3: Our proposed reward dimension reduction framework. We design a mapping function $f : \mathbb{R}^K \rightarrow \mathbb{R}^m$ from the original reward space to the reduced reward space.

As seen in Figure 3, we aim to design a mapping function $f : \mathbb{R}^K \rightarrow \mathbb{R}^m$ for reward dimension reduction, where $K > m \geq 2$. f transforms the original MOMDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, \mu_0, r, \gamma \rangle$ into another MOMDP $\mathcal{M}' = \langle \mathcal{S}, \mathcal{A}, P, \mu_0, f(r), \gamma \rangle$, reducing the dimensionality of the reward space while preserving essential features. We assume that standard multi-policy MORL approaches, such as Yang et al. (2019), perform adequately in the reduced-dimensional reward space of $\mathcal{M}'$. Then for any preference vector $\omega_m \in \Delta^m$, the $(m-1)$-simplex, the optimal multi-objective action-value function and the optimal policy are defined as:

$$Q^*_m(s, a, \omega_m) = \mathbb{E}_{P,\pi^*_m(\cdot|\cdot;\omega_m)|s_0=s,a_0=a}\left[\sum_{t=0}^{\infty} \gamma^t f(r_t)\right] \in \mathbb{R}^m, \quad \text{where} \tag{5}$$

$$\pi^*_m(\cdot|\cdot;\omega_m) = \arg\sup_{\pi} \omega_m^\top \mathbb{E}_{P,\pi}\left[\sum_{t=0}^{\infty} \gamma^t f(r_t)\right] = \arg\sup_{\pi} \mathbb{E}_{P,\pi}\left[\sum_{t=0}^{\infty} \gamma^t (\omega_m^\top f(r_t))\right]. \tag{6}$$

π^*_m is also expressed as $\pi^*_m(a|s, \omega_m) = 1$ if $a = \arg\max_{a'} \omega_m^\top Q^*_m(s, a', \omega_m)$, $\pi^*_m(a|s, \omega_m) = 0$ otherwise. Note that the key aspect of multi-policy MORL is that we learn the action-value function $Q^*_m(s, a, \omega_m)$ not just for a particular linear combination weight ω_m but for all possible weights $\{\omega_m \in \Delta^m\}$ in the training phase so that we can choose the optimal policy for any arbitrary combination weight depending on the agent's preference in the test or evaluation phase.

Our goal is to design a dimension reduction function, f, such that the policies learned in the reduced-dimensional space achieve high performance in the **original reward space** while satisfying two key requirements: (i) *online updates for dimension reduction* and (ii) *the preservation of Pareto-optimality* in the sense that $\forall \omega_m \in \Delta^m$,

$$\mathbb{E}_{\pi^*_m(\cdot|\cdot,\omega_m)}\left[\sum_{t=0}^{\infty} \gamma^t f(r_t)\right] \in CCS_m \Rightarrow \mathbb{E}_{\pi^*_m(\cdot|\cdot,\omega_m)}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \in \mathcal{F} \tag{7}$$

where CCS_m represents the convex coverage set in the reduced reward space and $\mathcal{F} \subset \mathbb{R}^K$ represents the Pareto frontier in the original reward space.

To the best of our knowledge, research on reward dimension reduction in MORL is limited, and there is no well-established evaluation protocol for this task. In this section, we propose a new training and evaluation framework tailored to the reward dimension reduction problem, along with the algorithm itself (outlined in Section 4.2). This framework facilitates a fair comparison of online dimension reduction techniques within the context of the original MOMDP.

During the *training phase*, we aim to learn the optimal multi-objective action-value function $Q_m^*(s, a, \omega_m)$ while we simultaneously update the dimension reduction function f online. For the action-value function update, we sample data (s, a, r, s') from a replay buffer but utilize the reduced-dimensional rewards $f(r)$ instead of the original rewards r. Our goal is to ensure that $\mathbb{E}_{\pi_m^*(\cdot|\cdot,\omega_m)}\left[\sum_{t=0}^{\infty} \gamma^t f(r_t)\right] \in CCS_m$ after the training phase ends.

In the *evaluation phase*, the learned policy $\pi_m^*(\cdot|\cdot, \omega_m)$ is tested on a set of N_e preferences $\Omega_{m,N_e} \subset \Delta^m$, with $N_e = |\Omega_{m,N_e}|$, where the points are evenly distributed on the $(m-1)$-simplex. For each $\omega_m \in \Omega_{m,N_e}$, we compute the expected cumulative reward $\mathbb{E}_{\pi_m^*(\cdot|\cdot,\omega_m)}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \in \mathbb{R}^K$ in the original reward space, as the MOMDP provides the high-dimensional vector reward at each timestep. Our goal is for the Pareto frontier of $\{\mathbb{E}_{\pi_m^*(\cdot|\cdot,\omega_m)}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \in \mathbb{R}^K | \omega_m \in \Omega_{m,N_e}\}$ to maximize hypervolume while minimizing sparsity.

4.2 DESIGN OF DIMENSION REDUCTION FUNCTION

To preserve the Pareto-optimality as shown in equation 7, we impose two minimal conditions on the dimension reduction function f:

Theorem 1. *If f is affine and each element of the matrix is positive, then equation 7 is satisfied.*

Proof. First, if f is **affine**, then $f(r) = Ar + b \in \mathbb{R}^m$, where $A \in \mathbb{R}^{m \times K}$. By linearity, $\forall \omega_m \in \Delta^m$, $\mathbb{E}_{\pi_m^*(\cdot|\cdot,\omega_m)}\left[\sum_{t=0}^{\infty} \gamma^t (Ar_t + b)\right] = A\left(\mathbb{E}_{\pi_m^*(\cdot|\cdot,\omega_m)}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]\right) + \frac{1}{1-\gamma} b \in CCS_m$.

Next, if each element of A is **positive**, we claim that $\mathbb{E}_{\pi_m^*(\cdot|\cdot,\omega_m)}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \in \mathcal{F}$ so that the Pareto-optimality in equation 7 is preserved.

This is proved by contradiction. Given $\omega_m \in \Delta^m$, suppose $\exists \pi' \in \Pi$ s.t. $\mathbb{E}_{\pi'}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] >_P \mathbb{E}_{\pi_m^*(\cdot|\cdot,\omega_m)}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$ in the original reward space. By the definition of $>_P$ in equation 1, each dimension of $\mathbb{E}_{\pi'}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] - \mathbb{E}_{\pi_m^*(\cdot|\cdot,\omega_m)}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \in \mathbb{R}^K$ is non-negative and at least one dimension is positive.

For each $1 \leq j \leq m$, let $a_j^\top \in \mathbb{R}^{1 \times K}$ be the j-th row vector of A and b_j be the j-th element of b. Then $a_j^\top\left(\mathbb{E}_{\pi'}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] - \mathbb{E}_{\pi_m^*(\cdot|\cdot,\omega_m)}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]\right) > 0$. In other words, we have $A\mathbb{E}_{\pi'}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] >_P A\mathbb{E}_{\pi_m^*(\cdot|\cdot,\omega_m)}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$ in the reduced-dimensional space. By linearity, adding $\mathbb{E}_{\pi'}\left[\sum_{t=0}^{\infty} \gamma^t b\right] = \mathbb{E}_{\pi_m^*(\cdot|\cdot,\omega_m)}\left[\sum_{t=0}^{\infty} \gamma^t b\right] = \frac{1}{1-\gamma} b$ to both sides gives

$$\mathbb{E}_{\pi'}\left[\sum_{t=0}^{\infty} \gamma^t (Ar_t + b)\right] >_P \mathbb{E}_{\pi_m^*(\cdot|\cdot,\omega_m)}\left[\sum_{t=0}^{\infty} \gamma^t (Ar_t + b)\right]. \tag{8}$$

Since CCS_m is by definition a subset of the Pareto frontier in the reduced-dimensional space, CCS_m consists of vector returns in the Pareto frontier. Therefore, equation 8 gives a contradiction since $\mathbb{E}_{\pi_m^*(\cdot|\cdot,\omega_m)}\left[\sum_{t=0}^{\infty} \gamma^t (Ar_t + b)\right] \in CCS_m$ is Pareto dominated by $\mathbb{E}_{\pi'}\left[\sum_{t=0}^{\infty} \gamma^t (Ar_t + b)\right]$. □

In short, to preserve Pareto-optimality in equation 7, we require the condition of $f(r) = Ar + b$ where $A \in \mathbb{R}_+^{m \times K}$. From equation 6,

$$\pi_m^*(\cdot|\cdot, \omega_m) = \sup_\pi \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t (\omega_m^\top A r_t)\right] + \frac{1}{1-\gamma} \omega_m^\top b = \sup_\pi \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t (\omega_m^\top A r_t)\right]. \tag{9}$$

In discounted reward settings, the bias term b does not affect the determination of π_m^*, so we set $b = 0$ for simplicity.

In addition, we impose another condition that A is **row-stochastic**: $\sum_{k=1}^{K} A_{jk} = 1, 1 \leq j \leq m$. Then

$$\sum_{k=1}^{K} (A^\top \omega_m)_k = \sum_{k=1}^{K} \sum_{j=1}^{m} A_{jk} (\omega_m)_j = \sum_{j=1}^{m} (\omega_m)_j \sum_{k=1}^{K} A_{jk} = \sum_{j=1}^{m} (\omega_m)_j = 1. \tag{10}$$

In other words, $\forall \omega_m \in \Delta^m$, we have the corresponding preference vector $A^\top \omega_m \in \Delta^K$ in the original reward space. Let $A^\top = [a_1, \cdots, a_m] \in \mathbb{R}_+^{Km}$ where $a_j \in \Delta^K, 1 \leq j \leq m$. Then $A^\top \omega_m \in \Delta^K$ and the set $\{A^\top \omega_m | \omega_m \in \Delta^m\} \subset \Delta^K$ is the convex combination of $a_j \in \Delta^K, 1 \leq j \leq m$. Conceptually, the role of the matrix A is to narrow down the preference search space from Δ^K to a proper subset of the Δ^K.

The next question is "how should we design the affine transform A to maximally preserve the information of the original vector reward function r?" To address this question, we propose constructing a reconstruction neural network g_ϕ, where the input is the reduced-dimensional reward $f(r)$. The network g_ϕ is trained to minimize the reconstruction loss:

$$\min_{A > 0,\ A \text{ row-stochastic}, \phi} \mathbb{E}_{s,a} \| r(s,a) - g_\phi(f(r(s,a))) \|^2 \tag{11}$$

where $A > 0$ denotes that each element of A is positive. This approach, combining compression with reconstruction, is widely employed to capture the essential features of input data while discarding irrelevant information (Baldi, 2012; Kingma & Welling, 2014; Berahmand et al., 2024). However, solving the optimization problem in equation 11 is more challenging than conventional autoencoder-style learning, where the encoder is a general neural network trained without constraints. In contrast, our method must ensure that the matrix A satisfies both the positivity constraint $A > 0$ and row-stochasticity during online training.

To overcome this challenge, we introduce a novel approach by parameterizing A using softmax parameterization, ensuring both positivity and row-stochasticity constraints are satisfied throughout training. Our implementation in PyTorch (Paszke et al., 2019) effectively applies this parameterization, and we solve the optimization in equation 11 using stochastic gradient descent in an online setting. The reconstruction loss is minimized alongside the training of the parameterized multi-objective action-value function $Q_\theta(s, a, \omega_m)$, which approximates $Q_m^*(s, a, \omega_m)$ as defined in equation 5.

Let $r(s,a) = [r_1(s,a), \cdots, r_K(s,a)]^\top \in \mathbb{R}^K$, where each r_k, for $1 \leq k \leq K$, lies in the scalar-valued function space $\mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$. The transformation $f(r)$ selects $m (< K)$ functions in $\mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$ based on a convex combination of $\{r_k\}_{k=1}^K$ within the reward function space. If the expressivity of g_ϕ is nearly perfect, we may achieve accurate reconstruction of $\{r_k\}_{k=1}^K$ even if f does not include all relevant information from r. To address this issue, we propose intentionally reducing the expressivity of g_ϕ by using dropout (Srivastava et al., 2014) to improve the generalization ability of f. Dropout was originally introduced to prevent overfitting, but as we show in the next section, this technique plays a key role in achieving diverse multi-objective solutions.

5 EXPERIMENTS

5.1 ENVIRONMENT AND BASELINES

While various practical applications require addressing many objectives (Fleming et al., 2005; Li et al., 2015; Hayes et al., 2022), there currently exist few MORL simulation environments with reward dimensions exceeding four (Hayes et al., 2022; Felten et al., 2023). To address this issue, we modified an existing traffic light control environment (Alegre, 2019) to create a sixteen-dimensional reward setting. Traffic light control is a practical example of a problem that can be formulated as MORL, where efficiently balancing many correlated objectives is crucial (Hayes et al., 2022).

As shown in Figure 4, the traffic intersection features four road directions (North, South, East, and West), each with four inbound and four outbound lanes. At each time step, the agent receives a state representing traffic flow information. The traffic light controller selects a phase as its action, and the reward is a 16-dimensional vector where each dimension corresponds to a measure proportional to the negative total waiting time of cars on the respective inbound lanes.

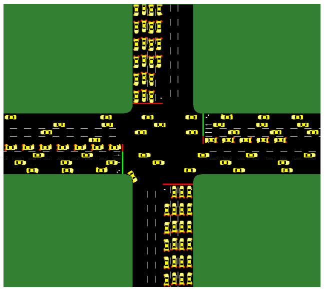

Figure 4: A snapshot of our considered environment: traffic light control.

In our experiments, we used the MORL algorithm from Yang et al. (2019) as the base algorithm for the original reward space (Base). We incorporated several online dimension reduction methods to the vector rewards in the base algorithm, including online autoencoder (AE) (Bank et al., 2020), incremental PCA (Cardot & Degras, 2018), our online implementation of conventional batch-based NPCA (Zass & Shashua, 2006), and our proposed approach. We followed the training and evaluation framework outlined in Section 4.1, setting $m = 16$ when evaluating the base algorithm alone.

For incremental PCA, we update the sample mean vector $\mu \in \mathbb{R}^K$ and the sample covariance matrix $C \in \mathbb{R}^{K \times K}$ at each timestep t using vector reward r_t. We periodically perform eigendecomposition on C and select the top m eigenvectors $u_1, \ldots, u_m \in \mathbb{R}^K$ corresponding to the largest eigenvalues, maximizing $\sum_{l=1}^{m} u_l^\top C u_l$. We construct the matrix $U = [u_1, \ldots, u_m] \in \mathbb{R}^{K \times m}$ so that $U^\top (r - \mu) \in \mathbb{R}^m$ represents the reduced vector for r, following the PCA assumption that the transformed vectors are centered (Cardot & Degras, 2018).

For the online variant of NPCA, we directly parameterize $U = [u_1, \ldots, u_m] \in \mathbb{R}^{K \times m}$ with a non-negativity constraint for efficient training (which we denote as $U \geq 0$). We optimize the objective $\max_{U \geq 0} \sum_{l=1}^{m} u_l^\top C u_l - \beta \|U^\top U - I_m\|^2$ using gradient descent, with a hyperparameter $\beta > 0$. Balancing the PCA loss with the orthonormality constraint creates a trade-off between capturing principal component information and maintaining orthonormal basis vectors. Both PCA and NPCA do not use reconstructor g_ϕ.

We set $m = 4$ for all online dimension reduction methods, as the sample covariance matrices consistently reached an effective rank of 4 by the end of training. The results presented are averages of hypervolume and sparsity over eight random seeds. (Additional implementation details can be found in Appendix A.)

5.2 RESULTS

	Base	PCA	AE	NPCA	**Ours**
HV(↑)	1.64×10^{62}	0	9.38×10^{61}	2.84×10^{62}	$\mathbf{1.51 \times 10^{63}}$
SP(↓)	4.41×10^{8}	7.47×10^{8}	7.54×10^{8}	5.39×10^{6}	$\mathbf{6.42 \times 10^{5}}$

Table 1: Performance comparison in our experiment where we set reference point for hypervolume evaluation to $(-10^4, -10^4, \cdots, -10^4) \in \mathbb{R}^{16}$. HV: hypervolume, SP: sparsity.

Table 1 demonstrates that our algorithm outperforms the baseline methods in both hypervolume and sparsity metrics. Specifically, our approach improves the base algorithm's hypervolume by a factor of 9.2 while significantly reducing sparsity, indicating that reward dimension reduction effectively scales the base algorithm to higher-dimensional spaces, resulting in more diverse and better-performing solutions.

The PCA-based dimension reduction is an affine transformation, but because the matrix does not meet the positivity condition in Theorem 1, it fails to guarantee Pareto-optimality as outlined in equation 7. Consequently, the PCA transformation moves the Pareto frontier points outside the desired region, leading to zero hypervolume. Similarly, the autoencoder method uses a nonlinear transformation that fails to satisfy the linearity requirement in Theorem 1, producing worse hypervolume and sparsity than the base case.

	NPCA	NPCA-ortho	**Ours**
HV(↑)	2.84×10^{62}	1.82×10^{60}	$\mathbf{1.51 \times 10^{63}}$
SP(↓)	5.39×10^{6}	9.83×10^{7}	$\mathbf{6.42 \times 10^{5}}$
Rank	1	4	4

Table 2: Performance comparison with NPCA and NPCA-ortho where "Rank" refers to the rank of the matrix in each method.

Our method outperforms NPCA by significantly increasing hypervolume and reducing sparsity, with improvements of 5.3x in hypervolume. Although NPCA employs an affine transformation with a nonnegative matrix, its online variant encounters instability due to the conflicting objectives of optimizing the principal component loss while maintaining the orthonormality constraint. As shown in Table 2, the best-performing NPCA models, in terms of hypervolume and sparsity, had matrices of rank 1. This indicates that the learning process prioritized maximizing the PCA loss, at the expense of enforcing the orthonormality constraint, producing completely overlapping basis vectors.

To address this, we tuned hyperparameters to emphasize the orthonormality constraint, denoting this variant as NPCA-ortho. However, Table 2 shows that this adjustment led to a performance decline compared to NPCA. The reason is that assigning more weight to the orthonormality constraint weakened the PCA update, significantly reducing its ability to capture relevant information from the original reward space. Additionally, we found that balancing the two losses was highly sensitive and difficult to fine-tune. In contrast, our method avoids these issues, offering a more stable and effective solution for reward dimension reduction without the trade-offs inherent in NPCA's design.

5.3 ABLATION STUDY

In this section, we explore the impact of the main components in our proposed dimension reduction approach. Specifically, we address two key questions:

1. How does limiting the expressivity of the reconstructor g_ϕ influence the diversity of the solutions obtained?
2. How do the conditions outlined in Section 4.2 affect the preservation of Pareto-optimality in equation 7?

	No dropout	**Ours**
HV(↑)	1.27×10^{63}	$\mathbf{1.51 \times 10^{63}}$
SP(↓)	2.73×10^{6}	$\mathbf{6.42 \times 10^{5}}$

Table 3: Ablation study on the effect of dropout during reward dimension reduction learning.

To answer the first question, we evaluated the effect of removing dropout during the learning of g_ϕ in equation 11. As shown in Table 3, omitting dropout resulted in a slight decrease in hypervolume, while sparsity sharply increased by a factor of 4.3. During online training, various preference vectors $\omega_m \in \Delta^m$ were sampled and corresponding vector rewards were collected. If the dimension reduction function f overfits to the initially collected reward data, the action-value function Q_θ may also overfit to a limited set of preference vectors, reducing the ability to acquire diverse solutions.

This highlights that limiting the expressivity of g_ϕ through dropout not only prevents overfitting of Q_θ but also forces the dimension reduction function to focus on extracting key features of the original reward space. In conclusion, dropout is critical for achieving a diverse set of solutions while maintaining a high hypervolume during online learning.

To address the second question, we analyzed the impact of the constraints applied to the dimension reduction function f. Specifically, we examined three aspects: bias, row-stochasticity, and positivity, to assess how these conditions influence the preservation of Pareto-optimality. Table 4 presents the results of our ablation study.

	Ours	+bias	-rowst	-rowst, -positivity
HV(↑)	$\mathbf{1.51 \times 10^{63}}$	1.23×10^{63}	5.88×10^{62}	0
SP(↓)	$\mathbf{6.42 \times 10^{5}}$	8.51×10^{5}	6.89×10^{6}	9.60×10^{8}

Table 4: Ablation study examining the impact of different conditions on the dimension reduction function f. "+bias" adds a bias term b in f; "-rowst" removes the row-stochasticity constraint while retaining the positivity condition; "-positivity" removes the positivity condition.

First, adding a bias term to f results in a slight decrease in hypervolume and an increase in sparsity compared to our method. However, the impact is less severe than the other modifications. While, in theory, the bias term b does not affect the determination of the optimal policy under discounted reward settings (as shown in equation 9), in practice, the finite episode length can introduce computational overhead. Thus, introducing a bias term offers minimal benefit, and a purely linear transformation is sufficient.

We next observe a performance drop when the row-stochasticity condition is removed, as shown in Table 4. Notably, sparsity increased sharply by a factor of 10.7, highlighting the detrimental impact of this removal. Note that the direction of each preference vector ω_m, not the magnitude, matters for the determination of optimal policy π_m^* in equation 6. By confining the search space to the simplex, the learning process can focus on finding the correct direction to extract essential reward information, rather than expending unnecessary effort on adjusting magnitudes. Consequently, enforcing the row-stochasticity constraint enhances learning efficiency, leading to more diverse solutions.

Finally, Table 4 demonstrates that using $f(r) = Ar$ with a generic linear matrix A fails to preserve Pareto-optimality, as outlined in equation 7. This is due to the lack of the positivity condition required by Theorem 1, which is analogous to the behavior observed with PCA in Section 5.2.

In conclusion, the positivity condition is essential for maintaining Pareto-optimality, while the row-stochasticity constraint plays a key role in improving the efficiency of online learning.

6 CONCLUSION AND FUTURE WORK

In this paper, we proposed a simple yet effective reward dimension reduction technique to address the scalability challenges of multi-policy MORL algorithms. By leveraging dimension reduction, our approach efficiently captures the key features of the reward space, enhancing both learning efficiency and policy performance while preserving Pareto-optimality during online learning. We also introduced a new training and evaluation framework tailored to reward dimension reduction, demonstrating superior performance in a sixteen-objective environment compared to existing methods.

While our approach offers a promising solution for scaling MORL algorithms, several avenues remain for future research. First, as discussed in Section 5.1, the lack of benchmarks for environments with more than ten objectives limits the comprehensive validation of our method. Developing robust benchmarks for high-dimensional MORL scenarios is a crucial direction for our future research.

Second, although we provided mathematical conditions for preserving Pareto-optimality in Theorem 1, these are only sufficient conditions. We investigated the effect of each condition in Section 5.3. However, our method's theoretical guarantees will be more solid if we establish the necessary conditions that pinpoint when Pareto-optimality fails.

Third, while our approach enables effective training for scalable MORL, for practical use, test preference vectors in their original high-dimensional form must be reduced to the lower-dimensional space learned by our model. Developing methods for preference vector reduction, and potentially integrating preference elicitation mentioned in Section 3, will be essential for making our approach more practical and complete.

Lastly, extending our method to constrained MORL represents a promising direction, especially for safety-critical tasks where additional constraints must be considered. This extension could open up new applications in areas where both optimality and safety are paramount.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of each algorithm in Section 5.1 and Appendix A, including the techniques, fine-tuned hyperparameters, and infrastructures used in our experiments. Upon acceptance, we will publicly release the source code to ensure accessibility and reproducibility. The evaluation protocol for performance comparison is outlined in Section 4.1. Additionally, Theorem 1 is presented in a self-contained manner, making it straightforward to verify the theoretical results.

REFERENCES

Axel Abels, Diederik M. Roijers, Tom Lenaerts, Ann Nowé, and Denis Steckelmacher. Dynamic weights in multi-objective deep reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 11–20. PMLR, 2019. URL `http://proceedings.mlr.press/v97/abels19a.html`.

Lucas N. Alegre. SUMO-RL. `https://github.com/LucasAlegre/sumo-rl`, 2019.

Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham W. Taylor, and Daniel L. Silver (eds.), *Unsupervised and Transfer Learning - Workshop held at ICML 2011, Bellevue, Washington, USA, July 2, 2011*, volume 27 of *JMLR Proceedings*, pp. 37–50. JMLR.org, 2012. URL `http://proceedings.mlr.press/v27/baldi12a.html`.

Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *CoRR*, abs/2003.05991, 2020. URL `https://arxiv.org/abs/2003.05991`.

Toygun Basaklar, Suat Gumussoy, and Ümit Y. Ogras. PD-MORL: preference-driven multi-objective reinforcement learning algorithm. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL `https://openreview.net/pdf?id=zS9sRyaPFlJ`.

Kamal Berahmand, Fatemeh Daneshfar, Elaheh Sadat Salehi, Yuefeng Li, and Yue Xu. Autoencoders and their applications in machine learning: a survey. *Artif. Intell. Rev.*, 57 (2):28, 2024. doi: 10.1007/S10462-023-10662-6. URL `https://doi.org/10.1007/s10462-023-10662-6`.

Hervé Cardot and David Degras. Online principal component analysis in high dimension: Which algorithm to choose? *International Statistical Review*, 86(1):29–50, 2018.

Hakan Ayhan Dagistanli and Özden Üstün. An integrated multi-criteria decision making and multi-choice conic goal programming approach for customer evaluation and manager assignment. *Decision Analytics Journal*, 8:100270, 2023.

Florian Felten, Lucas N. Alegre, Ann Nowé, Ana L. C. Bazzan, El Ghazali Talbi, Grégoire Danoy, and Bruno Castro da Silva. A toolkit for reliable benchmarking and research in multi-objective reinforcement learning. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*, 2023.

Peter J. Fleming, Robin C. Purshouse, and Robert J. Lygoe. Many-objective optimization: An engineering design perspective. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler (eds.), *Evolutionary Multi-Criterion Optimization, Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005, Proceedings*, volume 3410 of *Lecture Notes in Computer Science*, pp. 14–32. Springer, 2005. doi: 10.1007/978-3-540-31880-4_2. URL `https://doi.org/10.1007/978-3-540-31880-4_2`.

Matteo Giuliani, Stefano Galelli, and Rodolfo Soncini-Sessa. A dimensionality reduction approach for many-objective markov decision processes: Application to a water reservoir operation problem. *Environ. Model. Softw.*, 57:101–114, 2014. doi: 10.1016/J.ENVSOFT.2014.02.011. URL `https://doi.org/10.1016/j.envsoft.2014.02.011`.

Conor F. Hayes, Roxana Radulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M. Zintgraf, Richard Dazeley, Fredrik Heintz, Enda Howley, Athirai A. Irissappane, Patrick Mannion, Ann Nowé, Gabriel de Oliveira Ramos, Marcello Restelli, Peter Vamplew, and Diederik M. Roijers. A practical guide to multi-objective reinforcement learning and planning. *Auton. Agents Multi Agent Syst.*, 36(1): 26, 2022. doi: 10.1007/S10458-022-09552-Y. URL `https://doi.org/10.1007/s10458-022-09552-y`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL `http://arxiv.org/abs/1312.6114`.

John A Lee, Michel Verleysen, et al. *Nonlinear dimensionality reduction*, volume 1. Springer, 2007.

Bingdong Li, Jinlong Li, Ke Tang, and Xin Yao. Many-objective evolutionary algorithms: A survey. *ACM Comput. Surv.*, 48(1):13:1–13:35, 2015. doi: 10.1145/2792984. URL `https://doi.org/10.1145/2792984`.

Haoye Lu, Daniel Herman, and Yaoliang Yu. Multi-objective reinforcement learning: Convexity, stationarity and pareto optimality. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL `https://openreview.net/pdf?id=TjEzIsyEsQ6`.

Leland McInnes and John Healy. UMAP: uniform manifold approximation and projection for dimension reduction. *CoRR*, abs/1802.03426, 2018. URL `http://arxiv.org/abs/1802.03426`.

Giseung Park, Woohyeon Byeon, Seongmin Kim, Elad Havakuk, Amir Leshem, and Youngchul Sung. The max-min formulation of multi-objective reinforcement learning: From theory to a model-free algorithm. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=cY9g0bwiZx`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8024–8035, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html`.

Diederik M. Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *J. Artif. Intell. Res.*, 48:67–113, 2013. doi: 10.1613/JAIR.3987. URL `https://doi.org/10.1613/jair.3987`.

Diederik M. Roijers, Shimon Whiteson, and Frans A. Oliehoek. Linear support for multi-objective coordination graphs. In Ana L. C. Bazzan, Michael N. Huhns, Alessio Lomuscio, and Paul Scerri (eds.), *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, pp. 1297–1304. IFAAMAS/ACM, 2014. URL `http://dl.acm.org/citation.cfm?id=2617454`.

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

Umer Siddique, Paul Weng, and Matthieu Zimmer. Learning fair policies in multi-objective (deep) reinforcement learning with average and discounted rewards. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8905–8915. PMLR, 2020. URL `http://proceedings.mlr.press/v119/siddique20a.html`.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, 2014. doi: 10.5555/2627435.2670313. URL `https://dl.acm.org/doi/10.5555/2627435.2670313`.

Joshua B Tenenbaum, Vin de Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

Jie Xu, Yunsheng Tian, Pingchuan Ma, Daniela Rus, Shinjiro Sueda, and Wojciech Matusik. Prediction-guided multi-objective reinforcement learning for continuous robot control. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10607–10616. PMLR, 2020. URL `http://proceedings.mlr.press/v119/xu20h.html`.

Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 14610–14621, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/4a46fbfca3f1465a27b210f4bdfe6ab3-Abstract.html`.

Ron Zass and Amnon Shashua. Nonnegative sparse pca. *Advances in neural information processing systems*, 19, 2006.

Luisa M. Zintgraf, Diederik M. Roijers, Sjoerd Linders, Catholijn M. Jonker, and Ann Nowé. Ordered preference elicitation strategies for supporting multi-objective decision making. In Elisabeth André, Sven Koenig, Mehdi Dastani, and Gita Sukthankar (eds.), *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pp. 1477–1485. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018. URL `http://dl.acm.org/citation.cfm?id=3237920`.

A IMPLEMENTATION DETAILS

A.1 SOURCE CODE AND ENVIRONMENT

For our implementation, we adapted morl-baselines (Felten et al., 2023) and integrated it with sumo-rl (Alegre, 2019), a toolkit designed for traffic light control simulations, as discussed in Section 5. We will release our source code for public access upon acceptance.

The traffic light system offers four distinct phases: (i) Straight and right turns for North-South traffic, (ii) Left turns for North-South traffic, (iii) Straight and right turns for East-West traffic, and (iv) Left turns for East-West traffic. At each time step, the agent receives a 37-dimensional state, which includes a one-hot encoded vector representing the current traffic light phase, the number of vehicles in each incoming lane, and the number of vehicles traveling at less than 0.1 meters per second for each lane. The simulation starts with a one-hot vector where the first element is set to one. Based on this state, the controller chooses the next traffic light phase. The time between phase changes is 20 seconds, with each episode spanning 4000 seconds, or 200 timesteps. When transitioning to a different phase, the last 2 seconds of the interval display a yellow light to minimize vehicle collisions. The reward, represented by a 16-dimensional vector, is calculated as the negative total waiting time for vehicles on each inbound lane. The simulation runs for 52,000 timesteps in total.

A.2 BASELINES

For our proposed method and the baselines, we set the discount factor $\gamma = 0.99$ and use a buffer size of 52,000. In Base algorithm (Yang et al., 2019), we utilize a multi-objective action-value network Q_θ with an input size of $37 + K = 53$, two hidden layers of 256 units each, and ReLU activations after each hidden layer. The output layer has a size of $|\mathcal{A}| \times K = 4 \times 16 = 64$. For the dimension reduction methods, the Q_θ network has an input size of $37 + m = 41$, two hidden layers of 256 units with ReLU activations, and an output layer of size $|\mathcal{A}| \times m = 4 \times 4 = 16$.

We train Q_θ using the Adam optimizer (Kingma & Ba, 2015), applying the loss function after the first 200 timesteps, with a learning rate of 0.0003 and a minibatch size of 32. Exploration follows an ϵ-greedy strategy, with ϵ linearly decaying from 1.0 to 0.05 over the first 5,200 timesteps. The target network is updated every 500 timesteps. We update θ using the gradient $\nabla_\theta \mathcal{L}(\theta)$, $\mathcal{L}(\theta) = (1-\lambda)L^{\text{main}}(\theta) + \lambda L^{\text{aux}}(\theta)$, where $L^{\text{main}}(\theta)$ is the primary loss and $L^{\text{aux}}(\theta)$ is the auxiliary loss in Yang et al. (2019). The weight λ is linearly scheduled from 0 to 1 over the first 39,000 timesteps. Sampling preference vectors $\omega_m \in \Delta^m$ during training and execution follows the uniform Dirichlet distribution.

For the three online dimension reduction methods (our approach, the autoencoder, and our implementation of online NPCA), we utilize the Adam optimizer for updates. In our method, the matrix A is initialized with each entry set to $1/K = 1/16$. The neural network g_ϕ has an input dimension of $m = 4$, two hidden layers of 32 units each, and ReLU activations after each hidden layer. The output layer has a size of $K = 16$. We use a dropout rate of 0.75 (with 0 meaning no dropout). Equation equation 11 is optimized with a learning rate of 0.0003 and an update interval of 5 timesteps.

For the autoencoder, the encoder network has an input size of $K = 16$, two hidden layers with 32 units each, and ReLU activations after each hidden layer. The output layer has a size of $m = 4$. The decoder follows the same architecture as g_ϕ, but without dropout. The reward reconstruction loss is optimized with a learning rate of 0.0001 and an update interval of 20 timesteps.

For the online NPCA, we use ReLU parameterization for efficient learning (also implemented in PyTorch (Paszke et al., 2019)) to meet the constraint on matrix U. The matrix U is initialized similarly with each entry set to $1/K = 1/16$. NPCA is optimized with a learning rate of 0.0001, an update interval of 20 timesteps, and $\beta = 5 \times 10^4$. The reduced vector representation of r is $U^T(r-\mu) \in \mathbb{R}^m$, following the PCA assumption that the transformed vectors are centered (Zass & Shashua, 2006; Cardot & Degras, 2018). For NPCA-ortho, increasing the value of β did not yield better orthonormality, so we set the update interval to 5 timesteps, keeping the same β value.

For incremental PCA, we recursively update the sample mean vector of rewards as $\mu_{t+1} = \frac{t}{t+1}\mu_t + \frac{1}{t+1}r_{t+1} \in \mathbb{R}^K$ and the sample covariance matrix as $C_{t+1} = \frac{t}{t+1}C_t + \frac{t}{(t+1)^2}(r_{t+1} - \mu_t)(r_{t+1} - \mu_t)^\top \in \mathbb{R}^{K \times K}$ for each timestep t (Cardot & Degras, 2018). Every 20 timesteps, we eigen-decompose the covariance matrix, selecting the top m eigenvectors $u_1, \ldots, u_m \in \mathbb{R}^K$ corresponding to the largest eigenvalues, and update $U = [u_1, \ldots, u_m] \in \mathbb{R}^{K \times m}$. The reduced vector representation of r is $U^T(r-\mu) \in \mathbb{R}^m$, assuming the vectors are centered (Cardot & Degras, 2018). U is initialized as a matrix with each entry set to $1/K = 1/16$.

For evaluation, we set $N_e = 20$. For evenly distributed sampling and calculating hypervolume and sparsity, we use the implementation provided in Felten et al. (2023). We use infrastructures of Intel Xeon Gold 6238R CPU @ 2.20GHz and Intel Core i9-10900X CPU @ 3.70GHz.