Multi-Objective Utility Actor Critic with Utility Critic for Nonlinear Utility Function

Gao Peng*
Centrum Wiskunde
Informatica
Science Park 123,1098 XG Amsterdam
gao.peng@cwi.nl

Eric.J.Pauwels
Centrum Wiskunde
Informatica
eric.pauwels@cwi.nl

Hendrik Baier
Centrum Wiskunde
Informatica
TU/Eindhoven

Abstract

In multi-objective reinforcement learning (MORL), non-linear utility functions pose a significant challenge, as the two optimization criteria—scalarized expected return (SER) and expected scalarized return (ESR)—can diverge substantially. Applying single-objective reinforcement learning methods to solve ESR problems often introduces bias, particularly in the presence of non-linear utilities. Moreover, existing MORL policy-based algorithms, such as **EUPG** and **MOCAC**, suffer from numerous hyperparameters, large search spaces, high variance, and low learning efficiency, which frequently result in sub-optimal policies.

In this paper, we propose a new multi-objective policy search algorithm called Multi-Objective Utility Actor-Critic (MOUAC). For the first time in the field, MOUAC introduces a Utility Critic based on expected state utility to replace Q-value critic, value function, or distributional critic based on Q-values or value functions. To address the high variance challenges inherent in multi-objective reinforcement learning (MORL), MOUAC also adapts traditional eligibility trace to the multi-objective setting called MnES-return. Empirically, we demonstrate that our algorithm achieves state-of-the-art (SOTA) performance in on-policy multi-objective policy search.

1 Introduction

Modern reinforcement learning, particularly policy gradient methods, relies on the actor-critic paradigm: training both the actor and critic simultaneously, where the critic supports policy evaluation to guide the actor's policy improvement. While the most common choice historically has been value functions or Q-value functions, applying these in multi-objective reinforcement learning (MORL) introduces significant bias—especially when optimizing expected scalarized return (ESR) with nonlinear utility functions [7]. Reymond et al. [12] point out that for nonlinear utility cases, due to ESR and SER considerations, value functions or Q-values fail to accurately evaluate scalarized returns.

*Code could be open-sourced soon.

18th European Workshop on Reinforcement Learning (EWRL 2025).

One alternative, EUPG by Roijers et al. [14], applies scalarized returns to entire trajectories to handle ESR under nonlinear utilities. However, EUPG's drawback is that every step in the trajectory contributes equally, failing to distinguish good actions from bad ones. Another alternative, MOCAC by Reymond et al. [12], uses a distributional value critic, modelling the value function as a distribution (by category) to indirectly estimate expected utility. While MOCAC outperforms EUPG in some toy problems, modelling distributions in high-dimensional spaces is computationally expensive and introduces sensitivity to additional hyperparameters, such as clipping parameters and category choices, making MOCAC difficult to extend to more complex tasks.

In this paper, we propose a new on-policy, multi-objective, policy-based algorithm: Multi-Objective Utility Actor-Critic (MOUAC). Rather than modelling distributional value functions or equally applying scalarized returns, MOUAC directly leverages the difference between scalarized returns and expected state-utility functions, significantly reducing the cost from modelling value distributions. To address the uncertainty and variance arising from multi-objective Markov decision processes (MOMDPs) and their reward-conditioned structures, we extend eligibility trace [15] to the multi-objective ESR setting, introducing a utility-based trace called Mean n-step Expected Scalarized Return(MnES-return). We evaluate MOUAC through simulations and explain its behaviour within the framework of Generalised Policy Iteration. Empirically, MOUAC outperforms both MOCAC and EUPG; to the best of our knowledge, MOUAC achieves state-of-the-art performance on ESR tasks with nonlinear utility functions.

2 Preliminary

2.1 MORL

MOMDP Generally, MORL uses a Multi-Objective Markov Decision Process (MOMDP) to model the problem. MOMDP is represented by the tuple $<S, A, Tr, \gamma, \mu, \mathbf{r}>$, where S is the state space and A the corresponding action space, $Tr : S \times A \times S \to [0, 1]$ specifies the (probabilistic) transition kernel), $0 < \gamma \leq 1$ is a discount factor (we will use $\gamma = 1$ in this paper), Finally, $\mu : S \to [0, 1]$ characterises the initial state by specifying a probability distribution over all possible initial states, and $\mathbf{r} : S \times A \times S \to \mathbb{R}^d$ is a vector-valued reward function, specifying the immediate (possibly stochastic) reward for each of the considered $d \geq 2$ objectives.

Policy A policy π is mapping that in each state assigns a probability to each possible action: $\pi : S \times A \to [0, 1]$ The set of all possible policies is denoted by Π. Rather than a traditional memory-less policy, MORL often requires a reward-conditioned policy, which is defined similarly: $\pi : S \times \mathbb{R}^d \times A \to [0, 1]$.

Trajectory Assume that τ is a trajectory (path) through state space, generated by a policy π, starting at some s_0 and terminating in an absorbing state at time T. One way to describe this trajectory is by listing all the states, actions and rewards as a sequence:

$$\tau = \{s_0, a_0, \mathbf{r}_1, s_1, a_1, \mathbf{r}_2, s_2, \ldots, , s_{T-1}, a_{T-1}\mathbf{r}_T, s_T\}.$$

We will denote the total reward (return) accrued along path τ, up till, and including, time t as:

$$\mathbf{R}_t(\tau) = \sum_{k=1}^{t} \mathbf{r}_k(\tau) \tag{1}$$

and we will often write $\mathbf{R}(\tau)$ as shorthand for $\mathbf{R}_T(\tau)$.

Q value and Value function Simlar to single objective RL, Q value and value function could also help us evaluate a policy indirectly. Define the value of taking action a in state s under a policy , denoted as $Q^\pi(s, a)$: $Q^\pi(s, a) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{k=0}^{T} \gamma^k \mathbf{r}_k \mid s_0 = s, a_0 = a\right]$ we can compute the value of each state under that policy as the expected cumulative reward: $\mathbf{v}^\pi(s) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{k=0}^{T} \gamma^k \mathbf{r}_k \mid s_0 = s\right]$.

Pareto Dominance For two general vectors $\mathbf{x}, \mathbf{y} \in I\!R^d$ we say that $\mathbf{x}$ *Pareto-dominates* $\mathbf{y}$ (notation $\mathbf{x} >_p \mathbf{y}$) iff

$$\forall i = 1, \ldots, d : x_i \geq y_i, \quad \text{and} \quad \exists 1 \leq j \leq d : x_j > y_j.$$

Utlity Function Scalarization entails applying a *utility function* $u : I\!R^d \longrightarrow I\!R$ to the vector reward to obtain a scalar value. In many applications, the focus is on *linear* utility functions: $u(\mathbf{V}) = \sum_{i=1}^{d} w_i V_i = \mathbf{w}^T \mathbf{V}$, but in general, a utility function can be any function u that is strictly increasing in Pareto sense: $\forall \mathbf{x}, \mathbf{y} \in I\!R^d : \mathbf{x} >_P \mathbf{y} \Longrightarrow u(\mathbf{x}) > u(\mathbf{y})$. For simplicity, in this paper, we'll focus on the two-dimensional special case of the Leontief non-linear utility function($u(X) = \min(qx, y)$).

2.2 Maximising Expected Scalarized Returns (ESR)

Using the above notation we are interested in finding the policy π that maximises the **expected scalarized returns (ESR)**:

$$ESR: \qquad \max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[u(\mathbf{R}(\tau) \right]. \tag{2}$$

Notice how this optimisation problem differs from the more frequently encountered **scalarised expected returns (SER)** which aims to optimise

$$SER: \qquad \max_{\pi} u \left(\mathbb{E}_{\tau \sim \pi} \left[\mathbf{R}(\tau) \right] \right). \tag{3}$$

Unless the utility function u is linear, these two approaches yield different results, and ESR is more appropriate in applications where we need to optimise the utility of single policy rollout, rather than averaging over a large number of them [7, 11].

3 From Single Objective Policy Gradient to Multi Objective Policy Gradient for ESR

3.1 Recap of policy gradient theorem and (Advantage) Actor-Critic (A2C) algorithm

To motivate the approach taken in this paper we briefly recall how the policy gradient theorem gives rise to the (advantage) actor critic algorithm to optimise the expected (scalar) reward R. (Since this is only meant for illustrative purpuses, we focus on scalar rewards, and trivial utility $u(R) = R$). Specifically, suppose we have a policy π_θ that depends on a parameter θ and we need to optimise the expected (scalar) reward along trajectories, i.e. $\mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$. The required gradient is provided by the policy gradient theorem [17]:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[R(\tau) \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right] \tag{4}$$

The disadvantage of this formulation is every action a_t along the path τ is weighted with the same reward $R(\tau)$ which quantifies the total reward along the complete path. For this reason, it is standard to focus on the future reward (along the path) which is defined as:

$$R_t^+ := \sum_{k=t+1}^{T} r_k$$

and modify the gradient computation as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} R_t^+ \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right] \tag{5}$$

Clearly, for any given path, $q_\theta(s_t, a_t)$ would be a valid estimate of R_t^+ since τ takes action a_t in state s_t and adheres henceforth to policy π_θ. We therefore arrive at:

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} q_\theta(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right] \tag{6}$$

The final improvement follows from introducing the value function $v_\theta(s)$ as a baseline, thus reducing variance. This yields the A2C-version of the policy gradient theorem [17, 2]:

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}(q_\theta(s_t,a_t)-v_\theta(s_t))\,\nabla_\theta \log \pi_\theta(a_t\,|\,s_t)\right] \tag{7}$$

3.2 Multi-objective Policy Gradient for ESR

Policy Gradient has been extended to Multi Objective RL for quite a long time[10]. Reymond et al. [12] put forward EUPG and MOCAC to solve ESR based on policy gradient algorithms. To the best of our knowledge, EUPG and MOCAC are only existing policy gradient method aiming at ESR.

Objective Function of ESR Since we will focus on policy gradient methods, we recast eq. 2 for the case of a parametrised policy π_θ, which yields:

$$ESR: \qquad \max_\theta J(\theta) \qquad \text{where} \qquad J(\theta)=\mathbb{E}_{\tau\sim\pi_\theta}\left[u(\mathbf{R}(\tau))\right]. \tag{8}$$

EUPG The straightforward equivalent of the simple policy gradient eq. 4 is furnished in the EUPG algo where the policy gradient is given by the following equation:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[u(\mathbf{R}(\tau))\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t\,|\,s_t,\mathbf{R}_t)\right] \tag{9}$$

MOCAC[12] . For their algorithm, they require a multivariate distribution where $\vec{z_i}$ is a support atom for vectorial returns. V_{min}, V_{max} for each objective. Moreover, under the assumption that the same number of categories N for each objective-dimension, resulting in a discrete distribution with N^ocategories. Each atom $\vec{z_i}\forall i \in [\{1,2,3...N\}]^o$ then becomes: $\vec{z_i} = \vec{V_{min}} + i\Delta\vec{z}$ with $\Delta\vec{z} = \frac{\vec{V_{max}}-\vec{V_{min}}}{N-1}$. $V(s\_t)$ is computed in a similar manner as the single-objective case: $V(s\_t) = \sum_{i\in[\{1,2,3...N\}]^o}\vec{z_i}Z_\varphi(\vec{z_i}|s_t)$ where $Z_\varphi(\vec{z_i})$ is the associated probability. Then MOCAC use the gradient in way below with 1 step TD advantage:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi}\Bigg[\sum_{t=0}^{T}\Bigg(\sum_{i\in[\{1,2,3,\ldots,N\}]^o} u(R_{t+1}+\gamma^t\vec{z_i})Z_\varphi(\vec{z_i}) - \sum_{i\in[\{1,2,3,\ldots,N\}]^o} u(R_t+\gamma^t\vec{z_i})Z_\varphi(\vec{z_i})\Bigg)\nabla_\theta \log\pi_\theta(a_t|s_t,R_t^-)\Bigg]$$

4 MOUAC Methodology

4.1 Expected state utility, Expected state-action utility and utility critic

In this section we take the standard version of A2C (cf. eq. 7) and adapt it for ESR. We need to start by reformulating the state and state-action value functions, to include the effect of the utility function u. Since u is assumed to be non-linear, we can only estimate the value of states (and state-actions) if we include the accrued (vector) reward $\mathbf{R}$ (see eq. 1) upon arrival in that state. For this reason, we need to make the following modifications to the definitions of the value functions:

- **Expected state utility** determines the expected final utility under the policy π when starting in state s with the accrued cumulative reward $\mathbf{R}$:

$$v_\pi^u(s,\mathbf{R}) := \mathbb{E}_{\tau\sim\pi}\left[u\left(\mathbf{R}+\sum_{k=1}^{T}\mathbf{r}_k\right)\,|\,s_0=s,\mathbf{R}_0=\mathbf{R}\right] \tag{10}$$

- **Expected state-action utility** determines the expected final utility under the policy π when starting in state s with the accrued cumulative reward $\mathbf{R}$, and taking action a:

$$q_\pi^u(s,a,\mathbf{R}) = \mathbb{E}_{\tau\sim\pi}\left[u\left(\mathbf{R}+\sum_{k=0}^{T}\mathbf{r}_k\right)\,|\,s_0=s,\mathbf{R}_0=\mathbf{R},a_0=a\right] \tag{11}$$

Recall that due to the non-linearity of the utility function and ESR setting, the Bellman equation no longer holds. Extending TD-learning to update $q^u_\pi(s, R, a)$ as well as $v^u_\pi(s, R)$ remain to be explored in future. In this paper, we estimate them by Monte Carlo estimation with the scalarized episodic return.

The challenge with this A2C approach is to compute the **utility critic**, which is a utility based advantage

$$A^u = q^u - b^u \tag{12}$$

In general, any policy gradient estimator conforming to eq. 12 should give unbiased policy evaluation. In this equation, the baseline b^u is easily approached with $v^u(s_t, R_t)$ or $u(\tau)$, while q^u is not easy. Let's expand some options with regard to approximate q^u:

- $q^u \approx q^u(s_t, a_t, R_t)$ like in eq. 7. Directly use uitlity based A2C is most theoretically ideal but not practical. Multi-dimensional reward in the input space increases the complexity to approximate $q^u(s_t, a_t, R_t)$ in on-policy time. Further, q^u function applying a Temporal Difference learning way to evaluate the action is usually accompanied by high bias and low variance, especially in a stochastic environment.
- $q^u \approx u(R(\tau))$ is another natural choice. This means a Monte Carlo Estimation usually accompanying with high virance and low bias. In this paper, we use the mean square error from the scalarized episodic return to update the Expected state utility.

In the next section we introduce multi-objective utility actor critic (MOUAC) which we propose as a strategy to tackle the ESR optimisation.

4.2 KEY INSIGHT

Underpinning the MOUAC algorithm is the key insight that, through a change of point of view, the original ESR problem can be turned into regular single objective policy optimisation by extending the state space. Specifically, since the utility computation (in eqs. 10, 11) requires the accrued (vector) reward $\mathbf{R}_t$ when entering state s_t, we introduce the extended (reward-conditioned) states $s'_t := (s_t, \mathbf{R}_t)$, specifically $s'_t = (s_t, \mathbf{R}_t) \xrightarrow{a_t} s'_{t+1} = (s_{t+1}, \mathbf{R}_{t+1})$ where $\mathbf{R}_{t+1} = \mathbf{R}_t + \mathbf{r}_{t+1}$. In contrast to the original problem formulation, these extended states now contain all the information necessary to select (optimal) actions. Furthermore, since the accrued vector reward is now part of the state, we can formally introduce a new scalar reward that is zero for all non-terminal transitions, and equal to $u(\mathbf{R}_T)$ when transitioning to a terminal state.

One way to visualise this is that every roll-out path τ in the original state space $\mathcal{S}$ is "lifted" to a path τ' in the extended state space $\mathcal{S} \times \mathbb{R}^\Gamma$. It is easy to see that the value functions in this extended state space correspond the the utility-based values introduces in eqs. 10, 11): $v'_\pi(s'_t) = v^u_\pi(s_t, \mathbf{R}_t)$ and $q'_\pi(s'_t, a_t) = v^u_\pi(s_t, \mathbf{R}_t, a_t)$. This extended MDP can be viewed as a projection of the original MOMDP, and both share the same optimal policy. By moving to this extended space, the Bellman equation becomes valid again, which allows TD prediction to be applied in a REINFORCE-like framework. Concretely, when computing $q^u(s_t, \mathbf{R}_t, a_t)$ at time step t, this is equivalent to evaluating $q'(s'_t, a_t)$. In a standard RL setting, it is natural to approximate this quantity with TD bootstrapping rather than relying on a full Monte Carlo return. Because all intermediate transitions yield zero reward, the target can be expressed directly through the successor state, giving $q^u(s_t, \mathbf{R}_t, a_t) = q'(s'_t, a_t) \approx v'(s'_{t+n}) = v^u(s_{t+n}, \mathbf{R}_{t+n})$, where n denotes the horizon in n-step TD. This establishes how TD prediction can effectively substitute for Monte Carlo estimates within policy gradient updates when viewed through the extended MDP(details in appendix A.1).

4.3 Multi-Objective Utility Actor Critic (MOUAC)

MOUAC 1 is an on-policy algorithm that uses trajectories to estimate parametrised approximators π_θ and v^u_ω (parametrised by ω) for the optimal policy π and corresponding value function v^u respectively. The learning procedure could be described as repeating the following process until convergence: (1) use the current policy π_θ to roll out a trajectory τ; (2) update the policy function π_θ with utility critic eq. 15; (3) use $u(R(\tau))$ to update v^u_ω in a mean square error way. In MOUAC, we use a eligibility trace $q^u \approx \bar{\delta}(s_t, R_t)$ (see below for more details) to combine TD-learning and Monte Carlo estimation. We will discuss this in detail in section 4.4 below.

Algorithm 1 Multi-Objective Utility Actor Critic

Initialize: Policy π_θ, Utility Critic v^u_ω, learning rate α,
Repeat:
  Collect one trajectory τ using policy $\pi_\theta(a|s, \mathbf{R})$
  Compute policy gradient estimate15:
    $\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{t=0}^{T} \left(\bar{\delta}(s_t, \mathbf{R}_t) - v^u_\omega(s_t, \mathbf{R}_t)\right) \nabla_\theta \log \pi_\theta(a_t \mid s_t, \mathbf{R}_t)\right]$
  Update policy parameters :
    $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
  Update utility critic ω
  Increment episodes count, and clean the buffer
Until convergence

4.4 Mean n-step expected scalarized return(MnES-return)

While introducing a baseline can help lower variance, it is still insufficient on its own. From the perspective of credit assignment — the core challenge in policy evaluation — the advantage estimate we used above looks ahead n steps, which is equivalent to Monte Carlo estimation usually with high variance and low bias. More generally, to better manage the bias-variance trade-off in learning, we can modify the q^u term in equation eq. 12 by incorporating past evaluations(TD-learning).

The classical λ-return method offers one such approach: it introduces a parameter λ that balances attention between shorter and longer horizons. When λ is small, the algorithm emphasises short-term outcomes, similar to TD(0); as λ approaches 1, it behaves closer to TD(1) or Monte Carlo estimation, focusing more on long-term, full-trajectory returns. Building on this classical credit assignment framework, we derive two options for handling multi-objective reinforcement learning.

Let's recap the λ-return [17] at time t is defined as a weighted sum of all n-step returns:$R_t^{(\lambda)+} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)+}$ where the n-step return is looking n-step ahead before update:$R_t^{(n)+} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n-1} + \gamma^n V(s_{t+n})$. Alternatively, it can be computed recursively by advantage function using the temporal difference (TD) error[15], Generalised Advantage Estimation: $R_t^{(\lambda)+} = (r_t + \gamma V(s_{t+1}) - V(s_t)) + \gamma\lambda R_{t+1}^{(\lambda)+}$. However recall that, in our MOUAC algorithm, the Bellman equation does not hold in the non-linear ESR setting. These challenges mean that traditional n-step return and λ-return (or GAE) do not work in our case.

Inspired by the above work, here we could first try to consider the scalarised n-step return: at time step t, with accrued reward $\mathbf{R}_t$, look ahead n step, then update. We notice that a defined **expected state utility** could directly give us this evaluation because of its reward-conditioned nature. Then **scalarised n step return** at time step t could be directly represented with $v^u_\pi(s_{t+n}, R_{t+n})$. If we directly use this scalarised n step return and n is infinity, it means we use $u(R(\tau))$ to replace q^u.

Furthermore, we put forward a **mean n-step expected scalarized return(MnES-return)** :

$$\bar{\delta}(s_t, R_t) = \sum_{n=1}^{T-t} v^u_\pi(s_{t+n}, R_{t+n})/(T-t) \tag{13}$$

Similarly, Extending from GAE(lambda return), we could define **λ-step expected sclarized return(λ-ES-return)** at timestep t as

$$\delta^{(\lambda)}(s_t, R_t) = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} v^u_\pi(s_{t+n}, R_{t+n}) \tag{14}$$

which is a natural extension of λ-return from single objective to multi-objective. Here we uniformly use the credit assignment at time t to replace the q^u in eq12. So the final update equation comes to:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T} \left(\bar{\delta}_\omega(s_t, R_t) - v^u_\alpha(s_t, \boldsymbol{R}_t)\right) \nabla_\theta \log \pi_\theta(a_t \mid s_t, \boldsymbol{R}_t)\right]. \tag{15}$$

Of course, we could use equation14 or $V(s_{t+n}, R_{t+n})$ to replace the $\delta(s_t, R_t)$.

In later experiments, we empirically find that MnESR-return outperforms λ-ES-return.

4.5 why does MOUAC work?

Method	Policy Evaluation	Policy Improvement	objective
Policy Iteration	Exact V_π	Exact greedy π'	single objective
REINFORCE	MC return R_t	Gradient step	single objective
AC	TD or MC estimate of V_π	gradient step	single objective
A2C	Advantage estimate of $Q(s,a)$ and baseline	gradient step	single objective
PPO/TRPO	GAE estimation of A_π	Constrained policy updates	single objective
Q-learning	Q_π update via TD	Greedy w.r.t. Q	single objective
EUPG	Scalarized MC return $u(R_t)$	Gradient Step	MO:ESR
MOCAC	Advantage function over TD estimated distirbutional $Z(\vec{z} \mid s_t, R_t)$	Gradient Step	MO:ESR
MOUAC	Advantage estimate of MnESR-return estimation and baseline	Gradient step	MO:ESR

Table 1: GPI view for different policy gradient algorithms.

Sutton [17] noted that Generalized Policy Iteration (GPI) underpins nearly all reinforcement learning (RL) algorithms, framing them as iterative cycles of policy evaluation followed by policy improvement. In Table 1, we classify several popular algorithms from both RL and multi-objective RL (MORL) through the lens of GPI. Traditional RL methods, from Policy Iteration to PPO, focus on how a policy affects expected return — an approach insufficient for MORL settings. We argue that EUPG, MOCAC, and MOUAC successfully address episodic single-run (ESR) problems with nonlinear utility functions because they deliver accurate utility-based policy evaluation and perform policy improvement according to Eq. 12.

Here, based on the Policy Gradient Theorem [17], we present two simple lemmas to clarify the workings of MOUAC.

Lemma 1. *In MOMDP, the objective function:* $J_\theta = \mathbb{E}_{\tau\sim\pi_\theta}[u(\mathbf{R}(\tau)]$ *could be approximated by the gradient* $\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}(u(R(\tau)) - b(s_t, R_t))\nabla_\theta \log \pi_\theta(a_t|s_t, R_t)\right]$ *without bias.*

Lemma 2. *In MOMDP, whose objective function* $J(\theta) = q^u_\pi$*, let* τ *be a trajectory generated under policy* π_θ*, and* $\bar{\delta}(s_t, R_t) = \sum_{n=1}^{T-t} v^u_\pi(s_{t+n}, R_{t+n})/(T-t)$ *then:*

$$\nabla J(\theta) = E_{\tau\sim\pi,\mu}\left[\sum_{t=0}^{T}(\bar{\delta}(s_t, R_t) - b)\nabla \log \pi(a_t|s_t, R_t)\right]$$

is an unbiased estimation.

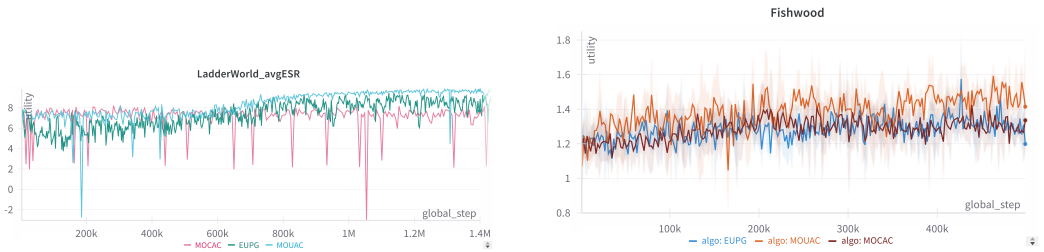
The proof for these lemmas could be found in appendix A.2 and A.3. Lemma 1 also has been proved by [12].

5 Simulation

Except in the simple policy section, we implement MOUAC, MOCAC, and EUPG using a three-layer MLP. MOUAC consistently outperforms MOCAC and EUPG on Fishwood and LadderWorld. To validate our EUPG and MOCAC implementations, we replicate the Fishwood setup from [12]. We also conduct an ablation study on MOUAC to assess the MnES-return component. Additionally, we test a one-parameter policy on a larger LadderWorld to gauge its gap from the optimal policy. For interested readers, additional details are provided in the appendix, along with two supplementary experiments on the Deep-Sea Treasure environment for evaluation. In these experiments, **MOUAC** continues to outperform the other methods.

5.1 LadderWorld

To test our algorithm in stochastic environment, we firstly test algorithms in the task called Ladder-World. The environment consists of two parallel rails connected by n rungs, forming a total of $2n$



(a) LadderWorld environment

(b) Fishwood under different conditions

Figure 1: MOUAC versus EUPG versus MOCAC

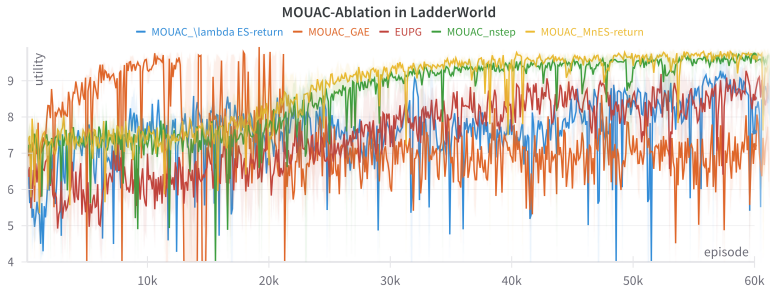


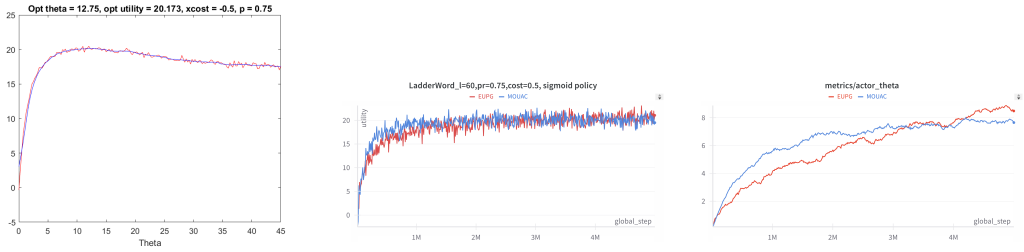
Figure 2: Ablation Study in LadderWorld: In a LadderWorld where length is 20, transition probability 0.9, cost 0.1, and the utility function is $u(X) = min(x_1, x_2)$. Run every algorithms 5 times and show its average result.

states. An initial state distribution μ determines the starting rail. At each state, the agent can choose between two actions: moving *forward* (F) along the current rail or *crossing* (C) to the opposite rail. Transitions are stochastic, with success probabilities p_F and p_C for forward and crossing actions, respectively. The reward vector depends on the actual (not the intended) outcome of the action. Specifically, moving forward along the top rail yields a reward $\mathbf{r} = (1, 0)$, while moving forward on the bottom rail yields $\mathbf{r} = (0, 1)$. Crossing between rails results in a penalty with reward $\mathbf{r} = -(\epsilon, \epsilon)$.

Fig. 1a shows the performance from MOCAC, EUPG and MOUAC. To notice, in the original MOCAC paper, MOCAC updates with the mini-batch in one epoch; except for MOCAC, other methods all update with one epoch of data in once. In the figure, MOUAC shows the best performance; MOCAC at the beginning converge to a not bad result, but later gets stuck at this level and becomes the worst algorithm; EUPG learns slowly, but outperforms MOCAC; MOUAC shows a stable learning curve than the others.

5.2 Fishwood

[12] suggests MOCAC outperforms EUPG in fishwood in Fig. 1b. We adapt the open-source code of MOCAC to the fishwood and keep the environment's parameters the same as using the following parameters: pf = 0.25,pw = 0.65,l = 13 where pf, pw, l are the probabilities of getting fish, wood, and the duration of an episode, respectively. Untiltiy function is $u = min\{fish, [wood/2]\}$. Our method MOUAC, outperforms EUPG(MOREINFORCE) and MOCAC. To note, the MOCAC result in the original paper is 1.4(since around 100k steps in their experiment), but in our simulation, only around 1.3(after 200k). This slight difference could come from the fine-tuning. This could support the argument that compared to MOCAC's complexity (parameters categories, value's bound, mini batch size, etc.), MOUAC with fewer parameters to tune is more friendly. Even compared to the best result of MOCAC in the original paper, MOUAC is as good as MOCAC, as MOUAC's utility is also 1.4[12].



(a) EU-AFO performance (b) Simple learning dynamics (c) Theta evolution

Figure 3: Experiment for ladder of length $n = 60$: TOP: Expected utility (based on Monte Carlo rollouts) for the 1-parameter policy π_θ specified in eq. 28 as a function of θ. For a crossing cost $xcost = -0.5$ and medium stochasticity $p_F = p_C = 0.75$, the optimal value $\theta^* \approx 12.75$ with a corresponding optimal utility $u^* \approx 20.17$. However, notice that the utility curve is relatively flat to the right of its maximum, hence it would be more accurate to state $8 \leq \theta^* \leq 15$. Results are based on 500 Monte Carlo rollouts per θ-value. Recall that for a deterministic environment the maximally achievable utility would be approximately 40. MIDDLE: Evolution utility $u(\mathbf{R}(\tau))$ under gradient ascent driven by EUPG (red, cf. eq. 9 and MOUAC (blue, cf. eq. 15). Notice how both methods reach the optimal utility $u^* \approx 20$ but MOUAC is more sample efficient. BOTTOM: Corresponding evolution of θ under the same dynamics. Notice how the dynamics stops in the neighbourhood of $\theta = 8$ where the utility hits a plateau (see TOP).

5.3 Ablation study MOUAC

Fig 2 contains the MOUAC with different eligibilty trace(to replace q^u in utility critic and b^u is always $v^u(s^t, R^t)$) and contains EUPG without b^u as a baseline. Among all the methode MOUAC_MnES-return13 outperforms all the other methods, and this is also what we use in alg.1. MOUAC_nstep with $u(R(\tau))$ minus baseline is slightly lower than MnES-return. EUPG is worse than MOUAC_nstep that support decreasing baseline is effective. EUPG and λ-ES-return converge to a similar result but EUPG is lower. GAE as our expectation give a bias compared to λ-ES-return.

5.4 Verify MOUAC optimality with simple policy

To test the proposed approach we compare EUPG to our algo for the following 1-parameter policy in the ladder world setting. The idea underpinning this policy is that the best action is the one that keeps the two components of the vector reward nearly balanced at all times: $qR_t^{(1)} \approx R_t^{(2)}$. To encode this idea in the policy we denote the imbalance between both outcomes as $z_t = R_t^{(2)} - qR_t^{(1)}$. Basically, the policy dictates to move forward, unless the difference z_t becomes too extreme. The switch point is determined by the parameter θ, and if we use $\sigma(x)$ to denote the standard sigmoid function (and $\overline{\sigma}(x) = 1 - \sigma(x)$), then we can write the policy as:

$$\pi_\theta(a = F \mid s, z) = \begin{cases} \sigma(z + \theta) & \text{if } s \text{ on top rail} \\ \overline{\sigma}(z - \theta) & \text{if } s \text{ on bottom rail} \end{cases} \tag{16}$$

Indeed, if we are on the top rail then we will want to move forward unless $qR^{(1)}$ exceeds $R^{(2)}$ by more than θ (i.e. $qR^{(1)} > R^{(2)} + \theta$, or equivalently, $z = R^{(2)} - qR^{(1)} < -\theta$). This behaviour is captured by the shifted sigmoid $\sigma(z + \theta)$, and a similar argument holds for the bottom rail. In Fig. 28, based on the above 1-parametric sigmoid function, MOUAC still slightly outperforms EUPG(speed) but ends with a lower θ. This latter obsevation could be the due to the flatness of the utility curve nears its maximum (see Fig. 3).

6 Related Works

Multi-Objective RL [13, 11] has quite long history study on finding pareto front without given utility and find policy with given utility. To reduce the reading load, we focus on the works that at include policy search method to solve MOMDP given utility function at least as a sub-problem. This will lead us to focus on the work either solve MOMDP with given utiltily or pareto front searching by

decomposition method[4]. Agarwal et al. [1] adapt a Multi Objective Actor Critic to nonlinear utility function problem but it use the linear utility function. [10, 9] has use policy based method to recover the continuous pareto front and they use linear utility function as well.

In the specific ESR setting of MORL given nonlinear utility function we are looking at, the solutions are quite limited at this moment. Roijers et al. [14] put forward EUPG the first algrithm to solve ESR. MOCAC[12] integrated C51[3] into EUPG so that it can approximate the expected utility based on the expected value distribution; this work is extremely suitable for risk-sensitive scenarios. To our humble knowledge, EUPG and MOCAC are the only policy based method aiming at ESR problem with non-linear utility function. The alternative way to solve this problem is MCTS variant: Hayes et al. [6] puts forward NLU-MCTS and Distributional Monte Carlo Tree Search(DMCTS) which learns a posterior distribution over the utility of the returns of a full episode.

In the past, researchers have developed many successful single-objective policy gradient methods, starting from REINFORCE, advancing to the Actor-Critic family[2, 8] that incorporates value baselines and temporal credit assignment, followed by stabilization techniques such as TRPO[15] and PPO[16], and more recently, improved exploration methods like Soft Actor-Critic[5]. Multi-objective reinforcement learning (MORL) has adapted these popular algorithms, leveraging advantage-style credit assignment for multi-objective scenarios. However, research on eligibility trace-based algorithms in this context remains rare, even though eligibility trace methods[15, 18] can help improve learning efficiency and reduce variance.

7 Conclustion

Given a nonlinear utility function, computing an optimal policy in multi-objective reinforcement learning (MORL) requires solving correlated Expected Scalarized Return (ESR) problems. Existing multi-objective policy-based methods face challenges such as sensitivity to parameters, difficulty in training, and low learning efficiency. This work introduces a new on-policy, policy-based approach called Multi-Objective Utility Actor-Critic (MOUAC) to address these issues. Unlike existing methods, MOUAC employs a utility critic and, for the first time, introduces an eligibility trace called MnES-return in the MORL setting to improve policy evaluation. As a result, MOUAC approximates state-of-the-art (SOTA) performance on ESR problems with nonlinear utility functions. We validate the effectiveness of the algorithm through simulations and provide a simple analysis explaining why MOUAC can solve multi-objective Markov decision processes (MOMDPs) more efficiently. The future direction could be extending MOUAC to offline MORL and MAMORL.

References

[1] Mridul Agarwal, Vaneet Aggarwal, and Tian Lan. Multi-objective reinforcement learning with non-linear scalarization. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 9–17, 2022.

[2] Mohammad Babaeizadeh, Iuri Frosio, Stephen Tyree, Jason Clemons, and Jan Kautz. Reinforcement learning through asynchronous advantage actor-critic on a gpu. *arXiv preprint arXiv:1611.06256*, 2016.

[3] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[4] Florian Felten, El-Ghazali Talbi, and Grégoire Danoy. Multi-objective reinforcement learning based on decomposition: A taxonomy and framework. *Journal of Artificial Intelligence Research*, 79:679–723, 2024.

[5] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[6] Conor F Hayes, Mathieu Reymond, Diederik M Roijers, Enda Howley, and Patrick Mannion. Distributional monte carlo tree search for risk-aware and multi-objective reinforcement learning.

In *Proceedings of the 20th international conference on autonomous agents and multiagent systems*, pages 1530–1532, 2021.

[7] Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):26, 2022.

[8] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[9] Simone Parisi, Matteo Pirotta, Nicola Smacchia, Luca Bascetta, and Marcello Restelli. Policy gradient approaches for multi-objective sequential decision making. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 2323–2330, 2014. doi: 10.1109/IJCNN.2014.6889738.

[10] Matteo Pirotta, Simone Parisi, and Marcello Restelli. Multi-objective reinforcement learning with continuous pareto frontier approximation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[11] Roxana Rădulescu, Patrick Mannion, Diederik M Roijers, and Ann Nowé. Multi-objective multi-agent decision making: a utility-based analysis and survey. *Autonomous Agents and Multi-Agent Systems*, 34(1):10, 2020.

[12] Mathieu Reymond, Conor F Hayes, Denis Steckelmacher, Diederik M Roijers, and Ann Nowé. Actor-critic multi-objective reinforcement learning for non-linear utility functions. *Autonomous Agents and Multi-Agent Systems*, 37(2):23, 2023.

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

[14] Diederik M Roijers, Denis Steckelmacher, and Ann Nowé. Multi-objective reinforcement learning for the expected utility of the return. In *Proceedings of the Adaptive and Learning Agents workshop at FAIM*, volume 2018, 2018.

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

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

[17] Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

[18] Hado van Hasselt, Sephora Madjiheurem, Matteo Hessel, David Silver, André Barreto, and Diana Borsa. Expected eligibility traces. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 9997–10005, 2021.

A Appendix

A.1 extended MDP from MOMDP

Here we define the extended MDP from 4.2 in detail.

Given the MOMDP $<S, A, Tr, \gamma, \mu, \mathbf{r}>$ with a nonlinear utility function $u(\vec{R})$, we want to optimise the ESR. Our target is to find a conditioned policy $\pi : S \times \mathbb{R}^d \times A \to [0, 1]$ that optimises the q^u and v^u functions we defined above.

Now we define a trajectory occupancy measure, conditioned on both state and accrued reward:

$$\sigma_\pi(\tau, s, r) = \mathbb{P}(\tau | s_0 = s, r_0 = r, \pi),$$

for this trajectory τ, that we use, $(s_T^\tau, \vec{R_T^\tau})$ to denote the last state of the trajectory; additionaly, T means the horizon of the trajectory; then we would have Terminal Set for this MOMDP with all the possible trajectories $\mathcal{T}\{(s_T^\tau, \vec{R_T^\tau})\}$

Then we could rewrite the eq 11 and eq 10 in such a way:

$$v_\pi^u(s, \vec{R}) = \sum \sigma_\pi(\tau, s, \vec{R})\, u(\tau_H) = \sum \sigma_\pi(\tau, s, R)\, u(\vec{R_T^\tau}))$$

and,

$$q_\pi^u(s, R, a) = \sum_{s_1, R_1} \mathbb{P}(s, R, a, s_1')\, v_\pi^u(s_1, R_1) \quad \text{where } s_1' = (s_1, R_1).$$

Now we build a define the extended MDP $M' = <S', A, Tr', \gamma, \mu', r'>$: State space $S' = S \times \mathbb{R}^d$, Action space is $A'(s, \vec{R}) = A(s)$, , Transition probability as $Tr' : \mathbb{P}(s_1' = (s_1, \vec{R} + \vec{r}(s, a))s_0' = (s_0, \vec{R}), a_0 = a) = \mathbb{P}(s_1 s_0, a)$ where $P(s_1|s_0, a)$ is the transition probability in the above MOMDP, discount factor γ , initial state probability μ', and reward function:

$$R'\big((s_0, \vec{R_0}), a, (s_1, \vec{R_1})\big) \;=\; \begin{cases} u(\vec{R_1}), & \text{if } s' \in \mathcal{T}, \\ 0, & \text{if } s' \notin \mathcal{T}, \end{cases} \qquad \text{where } \vec{R_1} = \vec{R_0} + r(s).$$

.

Then, in this MDP we will get the value function and q function as:

$$V_\pi(s, \vec{R}) \;=\; \sum \sigma_\pi(\tau, s, \vec{R}) * \sum R'(s_t', a, s_{t+1}') \tag{17}$$

$$\;=\; \sum \sigma_\pi(\tau, s, \vec{R}) * u(\vec{R_T^\tau}) \tag{18}$$

Here we notice,

$$V_\pi(s' = (s, \vec{R})) = v_\pi^u(s, \vec{R}) \tag{19}$$

In this sense, the MOMDP and its extended MDP share the same state space, transition dynamics, discount factor, and initial state distribution. With the specified reward-space transformation, they also yield the same objective function $J(\pi_\theta) = \mathbb{E}[u(R(\tau))]$. Consequently, the optimal trajectory distribution σ is identical in both formulations, and the optimal policy π^* must also coincide.

A.2 Lemma 1 Proof

Let's follow Policy Gradient Theorem[17] to prove that our MOUAC's policy improvement does not add bias to the original objective function.

Proof. The objective can be written as:

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}\left[u(\boldsymbol{R}(\tau))\right] = \nabla_\theta \int p_\theta(\tau)\, u(\boldsymbol{R}(\tau))\, d\tau.$$

Using the log-derivative trick:

$$\nabla_\theta p_\theta(\tau) = p_\theta(\tau) \nabla_\theta \log p_\theta(\tau),$$

we get:

$$\nabla_\theta J(\theta) = \int p_\theta(\tau)\, \nabla_\theta \log p_\theta(\tau)\, u(\boldsymbol{R}(\tau))\, d\tau = \mathbb{E}_{\tau\sim\pi_\theta}\left[u(\boldsymbol{R}(\tau))\nabla_\theta \log p_\theta(\tau)\right].$$

Assuming the environment dynamics are independent of θ, the policy is the only θ-dependent term in $p_\theta(\tau)$. Therefore,

$$\log p_\theta(\tau) = \sum_{t=0}^{T} \log \pi_\theta(a_t \mid s_t, \boldsymbol{R}_t),$$

and

$$\nabla_\theta \log p_\theta(\tau) = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t, \boldsymbol{R}_t).$$

Substituting back:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[u(\boldsymbol{R}(\tau)) \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t, \boldsymbol{R}_t)\right].$$

Now, introduce a baseline $b(s_t, \boldsymbol{R}_t, \tau)$ that is independent of a_t. Since its expectation over actions under the policy is constant with respect to θ, we can subtract it without bias:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \left(u(\boldsymbol{R}(\tau)) - b(s_t, \boldsymbol{R}_t, \tau)\right) \nabla_\theta \log \pi_\theta(a_t \mid s_t, \boldsymbol{R}_t)\right].$$

■ □

A.3 Lemma 2Proof

Proof. Here we start with the extended MDP equivalent to the given MDP. In the extended MDP, we use the policy gradient theorem with a baseline as the unbiased estimation $\nabla J(\theta)$.

$$\begin{aligned}
\nabla J(\theta) &= E_{\tau\sim\pi,\mu}\left[\sum_{t=0}^{T-1} (R'(\tau) - b)\nabla \log \pi(a_t|s'_t)\right] && (20)\\
&= E_{\tau\sim\pi,\mu}\left[\sum_{t=0}^{T-1} (Q(s'_t, a) - b)\nabla \log \pi(a_t|s'_t)\right] && (21)\\
&= E_{\tau\sim\pi,\mu}\left[\sum_{t=0}^{T-1} (R'(s'_t, a, s'_{t+1}) + V_\pi(s'_{t+1}) - b)\nabla \log \pi(a_t|s'_t)\right] && (22)\\
&= E_{\tau\sim\pi,\mu}\left[\sum_{t=0}^{T-1} (\sum_{i=0}^{n} R'(s'_t, a, s'_{t+i}) + V_\pi(s'_{t+n}) - b)\nabla \log \pi(a_t|s'_t)\right] && (23)
\end{aligned}$$

Eq 22 and Eq 23 above are separately 1-step TD and n-steps TD expansion from the original Actor-Critic. Because, $R'(s'_t, a, s'_{t+1}) = 0$ unless $s' \in \mathcal{T}$, then we could remove the R' above; n-step version would be:

$$\nabla J(\theta) = E_{\tau\sim\pi,\mu}\left[\sum_{t=0}^{T-1} (V_\pi(s'_{t+n}) - b)\nabla \log \pi(a_t|s'_t)\right] \quad (24)$$

Then, for any state we want to estimate, we could average all predictions for the successor states along the trajectory. Therefore, we have an equation like

$$\nabla J(\theta) = E_{\tau\sim\pi,\mu}\left[\sum_{t=0}^{T-1} (\frac{\sum_{n=1}^{T-t}(V_\pi(s_{t+n}, \vec{R}_{t+n}))}{T-t} - b)\nabla \log \pi(a_t|s'_t)\right] \quad (25)$$

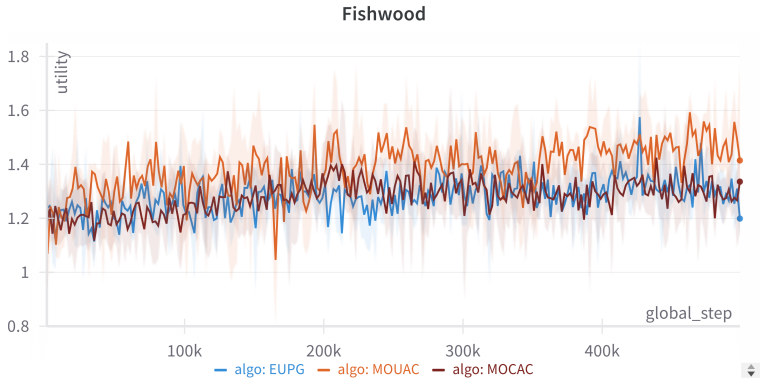


Figure 4: Explicit fig. 1

For we know eq 19, then we get

$$\nabla J(\theta) = E_{\tau\sim\pi,\mu}\left[\sum_{t=0}^{T-1}\left(\frac{\sum_{n=1}^{T-t}(v_\pi^u(s_{t+n},\vec{R}_{t+n}))}{T-t} - b\right)\nabla \log \pi(a_t|s_t,R_t)\right] \tag{26}$$

Then we get the estimator $\bar{\delta}$ is unbiased.

□

A.4 Fishwood

Fig. 4 shows an explicit version of Fishwood's result in Fig. 1. In this experiment, all the algorithms use a 2-layer MLP with 50 neurons in each layer to represent the actor and critic; the learning rate is 0.001, discounter factor 1. to specify, MOCAC's unique parameter: $c = 11$, $n_{step} = 9$, $v_{min} = (0,0)$, $v_{max} = (4,7)$,$clip\_grad\_norm = 50$. The fish and wood setting is below: : $pf = 0.25$,$pw = 0.65$,$l = 13$.

A.5 LadderWorld

Definition of the Ladder World (LW) environment To test the proposed methodology we introduce a simple environment that captures all the relevant aspect of the problem. Consider the following structured state space (see Fig. 5) which takes the shape of a ladder.

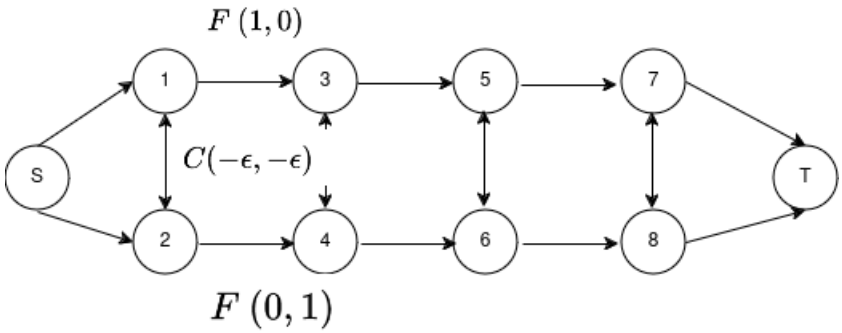


Figure 5: Ladder world: Each rail has n (regular) states (in this case, $n = 4$), in addition to the start state (S) and the absorbing terminal state (T). The arrows indicate the two possible actions in each (regular) state: `forward (F)` or `cross (C)`. Moving forward yields an immediate reward $\mathbf{r}_f = (1,0)$ or $(0,1)$ depending on whether you're on top or bottom rail. Crossing incurs a cost $\mathbf{r}_C = (-\epsilon,-\epsilon)$.

The ladder has two rails (top and bottom, say). **States** are located where rungs meet rails. There is a START state S and a terminal STOP state in which the final utility is computed. In the starting state, either the top or bottom rail is randomly chosen with equal probability. We will assume that there are n states on each rail and we use the following naming convention:

- Top rail: n states labeled: $1, 3, 5, \ldots 2n-1$ (odd state addresses)
- Bottom rail: n states labeled: $2, 4, 6, \ldots 2n$ (even state addresses)

The initial starting state S gets label 0, the absorbing terminal state gets label -1.

Actions, transitions, rewards, and utility

- In each (regular) state there are two **actions**: F (forward) and C (cross);
- For each action, the corresponding **transition** has a fixed and constant success rate. Specifically, let p_F be the probability that action F does indeed result in moving forward along the same rail, rather than crossing over (which therefore happens with probability $q_F = 1 - p_F$). Similarly, p_C is the success rate for the crossing transition. Notice that if $p_F = p_C = 1$, the transitions are deterministic.
- For each action (and corresponding transition) the **reward vector r** is determined solely by the actual transition. Moving forward along the same rail results in a reward of $\mathbf{r} = (1, 0)$ for the top rail and $\mathbf{r} = (0, 1)$ for bottom rail. Crossing incurs a slight cost $\mathbf{r} = (-\epsilon, -\epsilon)$.
- The total return $\mathbf{R}_T$ is the sum of all the incremental rewards $\mathbf{r}$. $\mathbf{R}_T$
- We will focus on the following **non-linear utility** for the final reward $\mathbf{R} = (R^{(1)}, R^{(2)})$:

$$u(\mathbf{R}) := \min(qR^{(1)}, R^{(2)}) \tag{27}$$

  where $q \geq 1$. In our experiments we take $q = 2$ which means that moving forward on the top rail is "twice as valuable" as moving forward on the bottom rail.

In addition to the neural network results presented in the main paper, we also employ one-parameter and two-parameter functions to represent the policy. This simplified formulation facilitates the identification of the optimal policy through numerical simulation. We evaluate **EUPG** and **MOUAC** under this setting, while **MOCAC** is excluded since it requires a neural network to predict categorical distributions.

One-parameter policy for the ladder problem To test the proposed approach we compare EUPG to our algo alg. 1 for the following 1-parameter policy in the ladder world setting. The idea underpinning this policy is that the best action is the one that keeps the two components of the vector reward nearly balanced at all times: $qR_t^{(1)} \approx R_t^{(2)}$. To encode this idea in the policy we denote the imbalance between both outcomes as $z_t = R_t^{(2)} - qR_t^{(1)}$. Basically, the policy dictates to move forward, unless the difference z_t becomes too extreme. The switch point is determined by the parameter θ, and if we use $\sigma(x)$ to denote the standard sigmoid function (and $\overline{\sigma}(x) = 1 - \sigma(x)$), then we can write the policy as:

$$\pi_\theta(a = F \mid s, z) = \begin{cases} \sigma(z + \theta) & \text{if } s \text{ on top rail} \\ \overline{\sigma}(z - \theta) & \text{if } s \text{ on bottom rail} \end{cases} \tag{28}$$

Indeed, if we are on the top rail then we will want to move forward unless $qR^{(1)}$ exceeds $R^{(2)}$ by more than θ (i.e. $qR^{(1)} > R^{(2)} + \theta$, or equivalently, $z = R^{(2)} - qR^{(1)} < -\theta$). This behaviour is captured by the shifted sigmoid $\sigma(z + \theta)$, and a similar argument holds for the bottom rail. Notice however, that since proceeding along the top rail is "twice as valuable" as doing so along the bottom rail (as $q = 2$), crossing behaviour should be different for the two rails.

The optimal value θ^* depends on the amount of stochasticity in the environment (i.e. p_F and p_C), as well as the crossing cost (ϵ). Obviously, if $\epsilon \approx 0$ then frequent crossing as little impact on the final utility and it pays to constantly balance both components of the reward (i.e. $\theta^* \approx 0$, especially in highly stochastic environments. By the same token , increasing the crossing cost ϵ will give rise to higher values for θ^*, as the policy will try and avoid costly crossings. The result is shown in Fig. 6

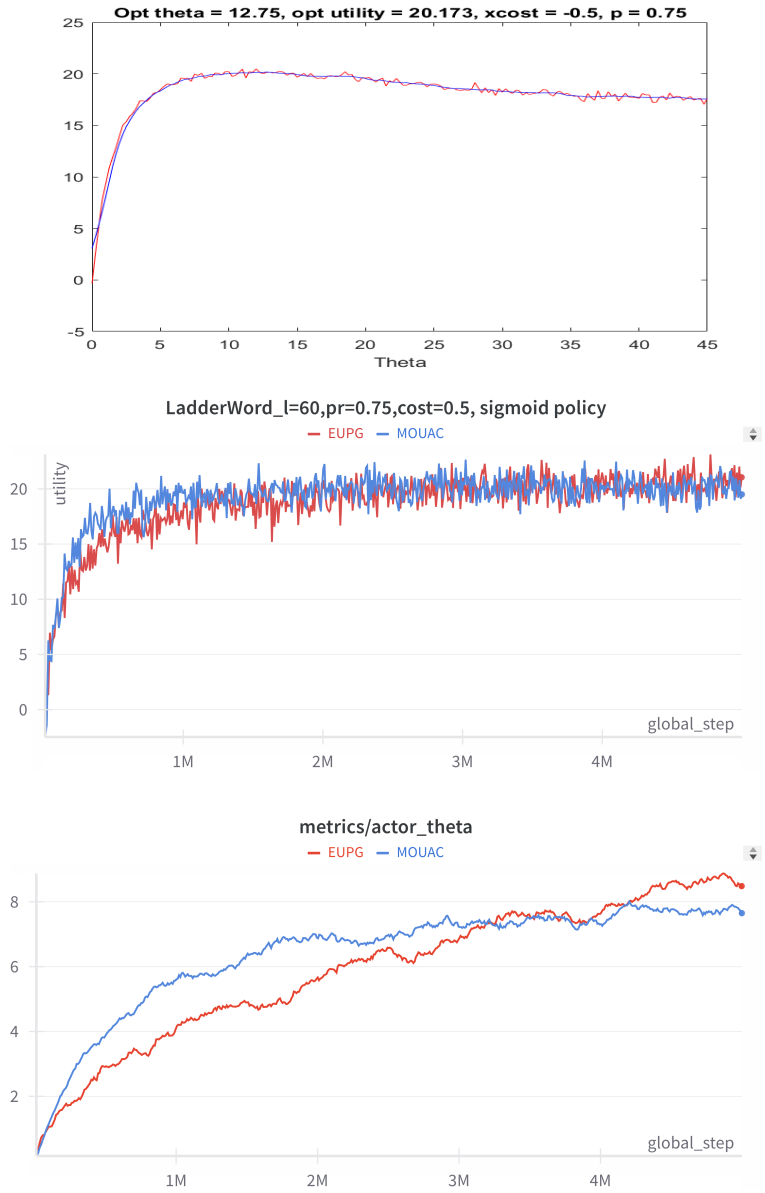


Figure 6: Experiment for ladder of length $n = 60$: TOP: Expected utility (based on Monte Carlo rollouts) for the 1-parameter policy π_θ specified in eq. 28 as a function of θ. For a crossing cost $xcost = -0.5$ and medium stochasticity $p_F = p_C = 0.75$, the optimal value $\theta^* \approx 12.75$ with a corresponding optimal utility $u^* \approx 20.17$. However, notice that the utility curve is relatively flat near its maximum, hence it would be more accurate to state $8 \leq \theta^* \leq 15$. Results are based on 500 Monte Carlo rollouts per θ-value. Recall that for a deterministic environment the maximally achievable utility would be approximately 40. MIDDLE: Evolution utility $u(\mathbf{R}(\tau))$ under gradient ascent driven by EUPG (red, cf. eq. **??**) and MOUAC (blue, cf. eq. 15). Notice how both methods the optimal $\theta^* \approx 20$ but MOUAC is more sample efficient. BOTTOM: Corresponding evolution of θ under the same dynamics. Notice how the dynamics stops in the neighourhood of $\theta = 8$ where the utility hits a plateau (see TOP).

Two-parameter Simple Policy Similarto one-parameter policy, algorithms are estimated by two parameters, a simple policy defined as:

$$\pi_\theta(F \mid n, s, \mathbf{R}^-, q) := \begin{cases} \overline{\sigma}(s - 2n\alpha) \cdot 1 + \sigma(s - 2n\alpha) \cdot \sigma(\gamma x) & \text{if } s \text{ on top rail 1} \\ \overline{\sigma}(s - 2n\beta) \cdot 1 + \sigma(s - 2n\beta) \cdot \sigma(\gamma x) & \text{if } s \text{ on bottom rail 2} \end{cases} \tag{29}$$

where γ as above, and $x = R_2^- - qR_1^-$.

The numerical simulation results are shown in Fig. 7, and the corresponding training results are presented in Fig. **??**. Fig. 7 illustrates the outcomes of the numerical simulation for a given environment. By executing the policy in Eq. 29 with systematically varied parameters α and β, we evaluate the resulting policies and construct the heatmap. The X-axis represents the values of α, while the Y-axis represents the values of β. Figures **??** and 8 respectively present the estimation results and the parameter variations for **EUPG** and **MOUAC**. From this analysis, we observe that **MOUAC** successfully identifies the optimal parameter configuration in this setting.

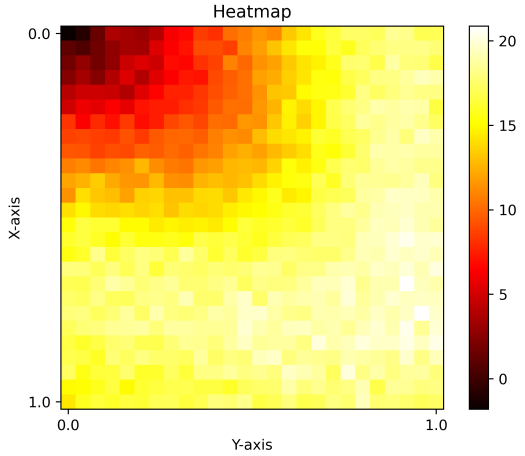
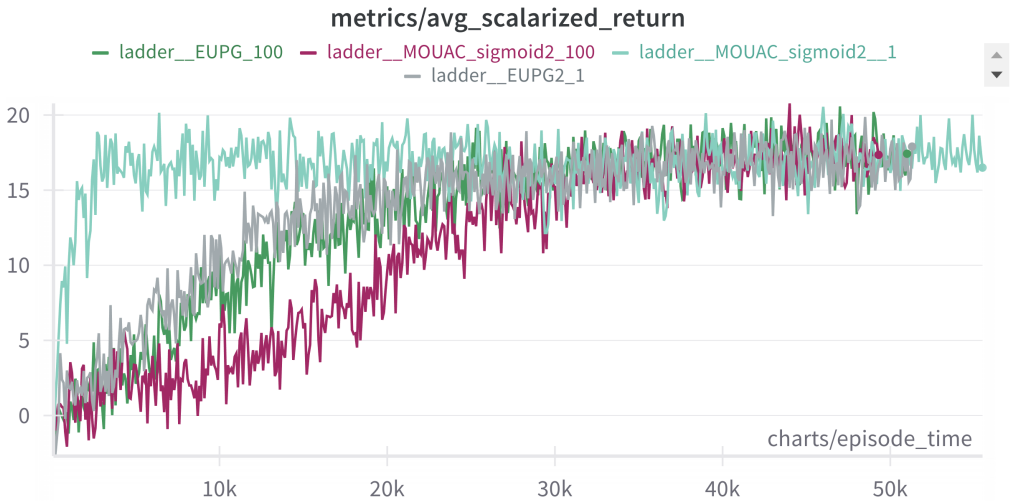
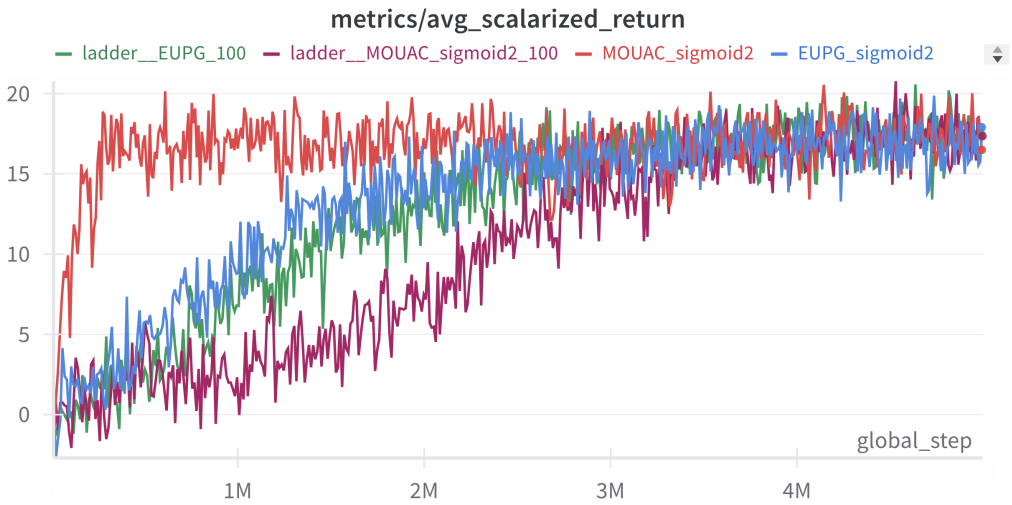


Figure 7: HeatMap to show the numerical simulation result.In ladder where length 60,$pf = pc = 0.75$ and $penalty = 0.5$ and with utility function $u(x, y) = min(2x, y)$.


metrics/avg_scalarized_return
ladder__EUPG_100
ladder__MOUAC_sigmoid2_100
ladder__MOUAC_sigmoid2__1
ladder__EUPG2_1
20
15
10
5
0
charts/episode_time
10k
20k
30k
40k
50k


metrics/avg_scalarized_return
ladder__EUPG_100
ladder__MOUAC_sigmoid2_100
MOUAC_sigmoid2
EUPG_sigmoid2
20
15
10
5
0
global_step
1M
2M
3M
4M

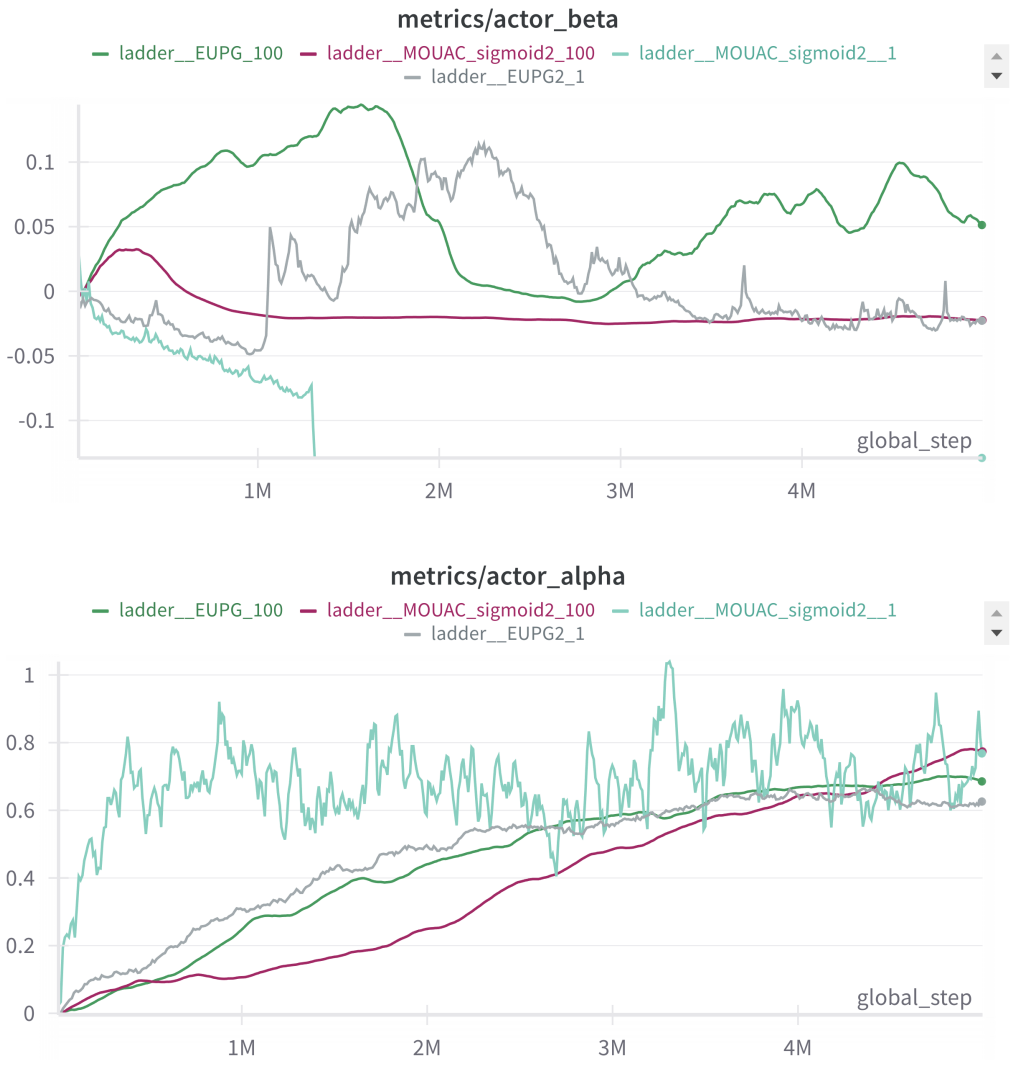


Figure 8: In the ladder shown in 7. We test EUPG and MOUAC with 2-parameter simple policy. The last part of each algorithms' name means how much episodes are collected before updating the policy.

A.6 Deep Sea Treasure

We compare **MOUAC**, **MOCAC**, and **EUPG** in the Deep-Sea Treasure environment (MOCAC's setting and MOGymnasium implementation), where the objective is to explore and collect the deepest treasures. The agent operates in a grid world, receiving higher treasure rewards at greater depths. Each move consumes one unit of oil, incurring a negative reward.

Repete experiments in deep-sea-treasure Moreover, we replicate the Deep-Sea Treasure experiment from the MOCAC paper [12]. In this setting, following their design, we represent the state as a one-hot vector over a 12×11 grid. The reward signal is defined as (r_0, r_1), where r_0 corresponds to the treasure obtained and r_1 denotes the oil consumed (treated as a positive quantity).

Their utility function is defined as: $f(x) = \begin{cases} \ln(1 + e^{r_0 - d_0}), & r_1 \leq d_1 \\ \ln(1 + e^{r_0 - d_0}) - (r_1 - d_1)^2 - p, & r_1 > d_1 \end{cases}$ where $d_0 = 45, d_1 = 10, p = 10$. The detailed results and experimental settings can be found in Figs. 6 and 8 of the MOCAC paper [12]. We adopt the same algorithmic parameters as specified for **MOCAC**: the actor is a three-layer MLP with architecture $[132, 50, 4]$, while the critic is a four-layer MLP with architecture $[132, 50, 50, 121]$ (with $c = 11$). The value bounds are set to $V_{\min} = (0, -20.1)$ and $V_{\max} = (0, 100.1)$.

Slightly deviating from their hyperparameter configuration, we tune a few general parameters for stability: the learning rate is set to 0.005, with $\gamma = 0.95$, $v_{\text{coef}} = 0.1$, and $\text{entropy}_{\text{coef}} = 0.1$. The settings for **EUPG** and **MOUAC** remain as described earlier, and both demonstrate relatively stable training dynamics.

As shown in Fig. 9, we evaluate performance across 10 independent runs. Due to the sparse reward structure of this task, all algorithms occasionally become trapped in sub-optimal saddle points. Overall, however, **MOUAC** consistently outperforms both **MOCAC** and **EUPG**.

To reproduce the original experiments, we adapted the authors' implementation to our framework. Since **MOCAC**'s original dependencies (e.g., legacy versions of `gym`) are no longer readily available online, we ported their algorithm into our environment, in a manner similar to `MORL_baseline`. Nevertheless, we observed that **MOCAC**'s training is less stable and more sensitive to hyperparameters, likely due to the combined challenges of deep exploration and parameter tuning. In particular, we found **MOCAC** to be highly sensitive to the random seed in PyTorch, leading to non-negligible variability in outcomes. As our focus is on mean estimators rather than distributional estimators, we leave a more systematic investigation of **MOCAC**—especially regarding exploration mechanisms such as entropy regularization and clamping technique.

deeper exploration in deep-sea-treasure task In Fig. 10, we compare **MOUAC**, **MOCAC**, and **EUPG** in the Deep-Sea Treasure environment (MOGymnasium implementation), where we adopt a nonlinear utility function

$$u(x, y) = \min\big(x,\ 0.5 \cdot (y + 100)\big), \tag{30}$$

where the first dimension (x, treasure) is prioritized. The bias of 100 in the second dimension effectively offsets oil consumption, transforming it into a positive term that encourages deeper exploration. To explain, the reason why we do not extend the MOMCAC's utility function is except this Leontief Function is also used in other experiments, it is also convenient to avoid the gradient explosion brought by high utility.

The agent's neural network input consists of its coordinates (2 dimensions) and the accumulated reward (2 dimensions). Both the actor and critic are parameterized as two-layer MLPs with architecture $[50, 50]$ and $\tanh$ activations.

Under this utility formulation, a more effective algorithm should discover treasures at greater depths. As shown in Fig. 10, **EUPG** tends to remain stuck at shallow treasures (around 8.2), while **MOCAC** converges at an intermediate level (~ 16). In contrast, **MOUAC** consistently outperforms both, attaining values around 23.2. Notably, the superiority of MOCAC over EUPG is consistent with the findings of [12], though in contrast to their reported results on the LadderWorld environment.

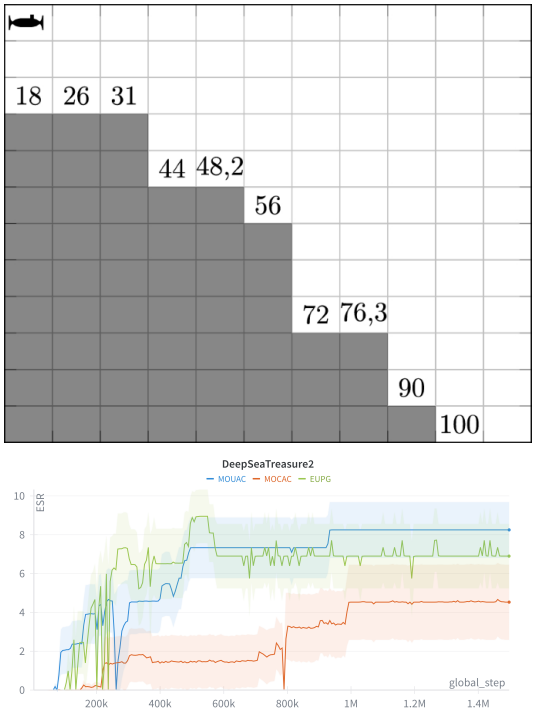


Figure 9: The environment in MOCAC[12] and our experiments with 10 runs. The line curve shows the mean ESR and its std error. In this setting, the optimal utility should happen in (56,10), which utility is around 10; (44,7) is 0.3 and (48,8) is 0.8.

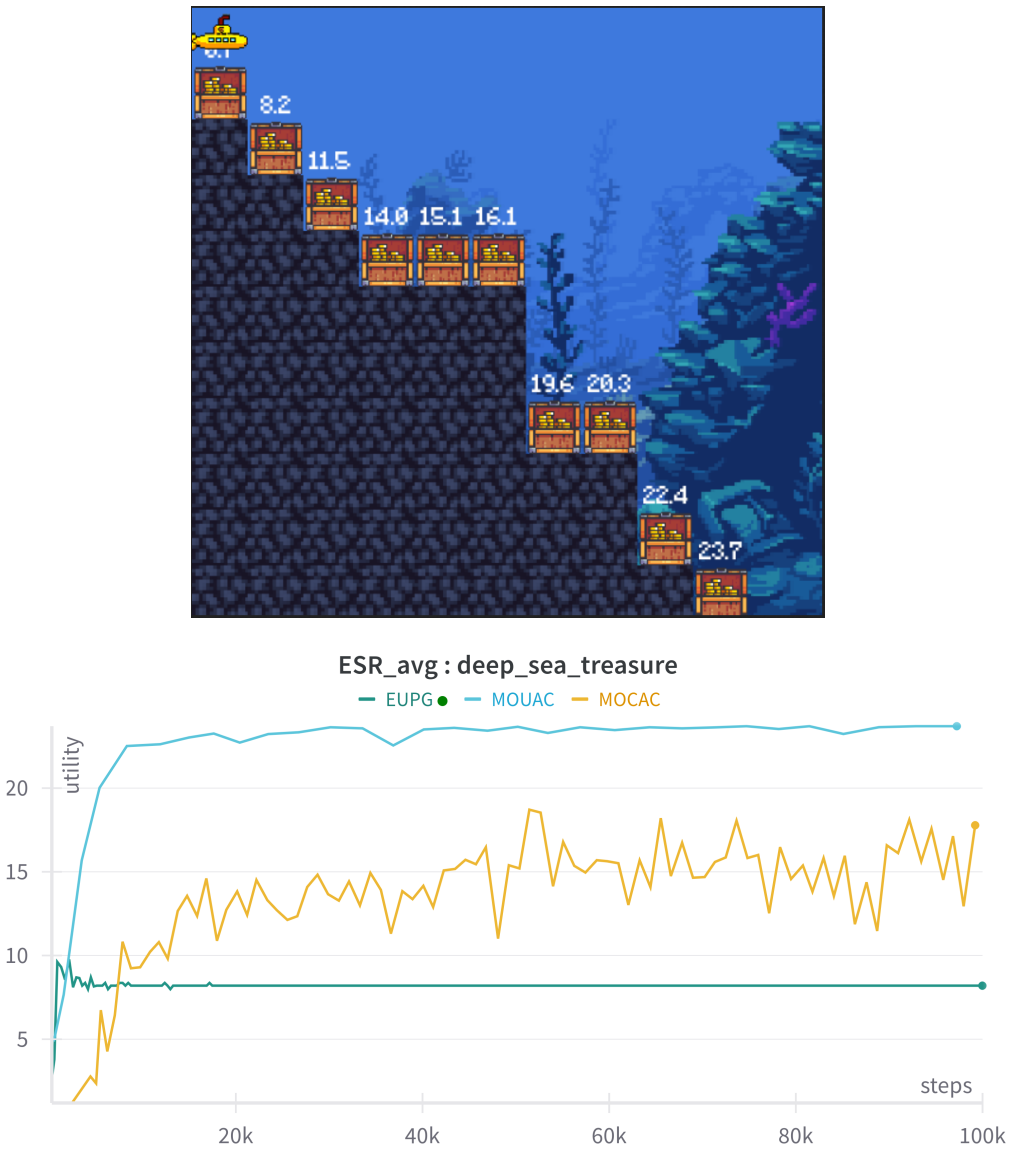


Figure 10: Deep-Sea Treasure results. Using the nonlinear utility function

$$u(x, y) = \min\big(x,\, 0.5 \cdot (y + 100)\big),$$

the agent primarily prioritises the first dimension (treasure). Under this formulation, **EUPG** consistently gets stuck at shallow treasures, **MOCAC** achieves intermediate depths, while **MOUAC** reliably discovers the deepest treasure.