BiES: Adaptive Policy Optimization for Model-based Offline Reinforcement Learning[⋆]

Yijun Yang[1,2], Jing Jiang[1], Zhuowei Wang[1], Qiqi Duan[2], and Yuhui Shi[2]

[1] AAII, University of Technology Sydney, Ultimo NSW2007 Sydney, Australia
{yijun.yang-1,zhuowei.wang}@student.uts.edu.au
jing.jiang@uts.edu.au

[2] Department of Computer Science and Engineering, Southern University of Science and Technology, 518055 Shenzhen, China
11749325@mail.sustech.edu.cn, shiyh@sustech.edu.cn

Abstract. Offline reinforcement learning (RL) aims to train an agent solely using a dataset of historical interactions with the environments without any further costly or dangerous active exploration. Model-based RL (MbRL) usually achieves promising performance in offline RL due to its high sample-efficiency and compact modeling of a dynamic environment. However, it may suffer from the bias and error accumulation of the model predictions. Existing methods address this problem by adding a penalty term to the model reward but require careful hand-tuning of the penalty and its weight. Instead in this paper, we formulate the model-based offline RL as a bi-objective optimization where the first objective aims to maximize the model return and the second objective is adaptive to the learning dynamics of the RL policy. Thereby, we do not need to tune the penalty and its weight but can achieve a more advantageous trade-off between the final model return and model's uncertainty. We develop an efficient and adaptive policy optimization algorithm equipped with evolution strategy to solve the bi-objective optimization, named as BiES. The experimental results on a D4RL benchmark show that our approach sets the new state of the art and significantly outperforms existing offline RL methods on long-horizon tasks.

Keywords: Offline reinforcement learning · Multi-objective optimization · Evolution strategy.

1 Introduction

Reinforcement learning (RL) encounters many obstacles when deploying an agent to real-world tasks, e.g., autonomous driving [22], robot control [21], and healthcare [27], due to costly online trial-and-error. Fortunately, it is available for these tasks to pre-collect large and diverse datasets. Hence, the research on learning high-quality policies from static datasets has promoted the development of offline RL [15].

Since the entire learning process is carried out in a static dataset D, offline RL faces several challenging problems. (1) RL agents cannot explore environments: If D does not

[⋆] This work is partially supported by the Shenzhen Fundamental Research Program under the Grant No. JCYJ20200109141235597 comprise highly rewarding demonstrations, the RL algorithms might be unable to learn a satisfying policy. Hence, the static dataset should be as large and diverse as possible [8]. (2) Another fundamental challenge is the distribution shift: The RL algorithms train a candidate policy on the distribution of data different from the distribution visited by the behavior (data collection) policy, which yields function approximation errors and results in poor performance. Several techniques have been proposed in response to the problem, such as behavior policy regularization [9] and Q-network ensembles [14]. These works in offline RL mainly focus on model-free methods. However, the recent work by [28] finds that even a vanilla model-based RL (MbRL) method can outperform model-free ones in the offline setting.

Model-based RL commonly learns an approximated dynamics model of a real Markov decision process (MDP), according to previously collected data. This paradigm benefits from powerful supervised learning techniques, allowing the learning process to leverage large-scale datasets. Moreover, once the MDP model is learned, we can employ it to generate trajectories resulting from applying a sequence of actions. As a result, MbRL algorithms have higher sample-efficiency than model-free ones. Despite these benefits, MbRL may suffer from the effect of the distribution shift issue [5] when using offline datasets. In particular, since offline datasets are unlikely to traverse all state-action pairs, the learned MDP model may not be globally accurate. Policy optimization using the model without any precautions against model inaccuracy can lead to the model exploitation issue [11], resulting in poor performance. For instance, the policy is likely to visit *risky* states where the model erroneously predicts successor states that yield higher rewards than the correct successor states obtained from the corresponding real MDP environment. One commonly-used way [28] of solving the issue is to incorporate uncertainty quantification into the model reward: $\tilde{r} = \hat{r} - \lambda u(s, a)$, which provides the agent with a penalty for visiting *risky* states.

However, it can be difficult to design a proper penalty term for complicated constraints [26]. In particular, The effort required to tune the reward penalty to a given offline RL task or repeatedly calculate it during optimization might negate any gains in the eventual model return. Conversely, in the case of a deficient penalty, a much larger region will be searched in a potentially risky policy space, resulting in an extra cost of time and unstable performance. On the other hand, recent works [11,28,13] have adopted existing RL algorithms, such as SAC [10] or NPG [19] to optimize a policy under a learned MDP model. In general, these algorithms are sample efficient due to learning the policy from every time step of an episode. However, in the offline model-based setting, such a learning process may damage the performance of those algorithms since model errors rapidly accumulate with the increase of the time steps, especially

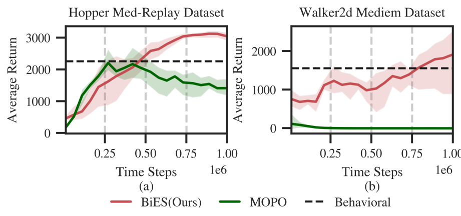

Fig. 1: **A proof-of-concept experiment on two offline RL tasks from the D4RL benchmark [8].** We evaluate a SoTA offline MbRL algorithm named MOPO. The results show that it cannot achieve a higher true return than the behavior policy when using long-horizon model rollouts.

when the length of the episode is long [5]. As a proof-of-concept experiment in Figure 1, we evaluate a state-of-the-art offline MbRL algorithm named MOPO [28]. Even though MOPO applies a well-designed reward function, the results show that it cannot achieve a higher true return than the behavior policy when using long-horizon model rollouts. This finding corroborates the combination of the uncertainty penalty and RL algorithms potentially not being a panacea to the model-based offline RL tasks, which motivates two effective improvements to existing approaches in this paper. Our contributions include the following:

(1) We propose a bi-objective policy optimization algorithm where the first objective aims to maximize the model return, and the second objective synchronously calibrates the learning bias of the policy. Our method achieves more stable policy improvement on offline MbRL tasks.
(2) To the best of our knowledge, our approach is the first to adopt evolution strategy (ES) to model-based offline RL problems and solve the optimization under uncertain and long-horizon RL tasks. We also theoretically establish an upper bound for the norm of a BiES-based gradient estimation.
(3) We conduct a large-scale empirical study on offline MuJoCo locomotion tasks from the D4RL benchmark [8]. The experimental results show that our method attains state-of-the-art results compared to other offline RL algorithms.

2 Related Work

Model-based Offline RL Although it offers the convenience of working with large-scale datasets, the MbRL algorithm still suffers from the effects of the distribution shift, especially in the model exploitation problem [15]. Prior works in MbRL algorithms explored methods to solve this problem, such as Dyna-style algorithms [23,11], the leverage of multiple dynamics models as an ensemble [5,11], an energy-based model regularizer [2], a game-theoretic MbRL algorithm [19], meta-learning [6], policy constraints [1], and generative temporal difference learning [12]. In the offline setting, since the learned model will not be calibrated with additional data collection, it becomes crucial to prevent the policy from overly exploiting the model. Recent works propose an explicit reward penalty for this purpose [13,28]. One constructs terminating states based on a hard threshold, and the other uses a soft reward penalty associated with a user-chosen weight.

Evolution Strategy in RL As a sub-class of the evolutionary algorithm (EA), we have seen a specific revival in evolution strategy (ES) on account of its surprising scalability and performance [20]. More particularly, recent works have applied ES to solve high-dimensional RL problems, resulting in the achievement of comparable performance to deep RL algorithms while cutting down the training time [20,17]. However, these works primarily focus on model-free tasks [4]. In the EA community, the works similar to our approach are model-based EA [3] and surrogate model-assisted EA [16], but these methods are exploited mainly in one-step black-box optimization problems rather than sequential RL tasks. Prior works showed that ES might be a more suitable choice for optimization under uncertain and long-horizon RL tasks because it learns from the result of the whole rollouts [20,17].

3 Preliminaries

We now describe the background related to our approach, including a baseline for offline MbRL tasks and the basic concepts of evolution strategy.

3.1 Model-based Offline Policy Optimization

A Markov Decision Process (MDP) is defined as $\mathcal{M} = \{S, A, r, p, \rho_0\}$, where S is the state space; A is the action space; r defines the reward function $S \times A \to \mathbb{R}$; p is the transition distribution $p(s_{t+1}|s_t, a_t)$; and ρ_0 is the probability distribution of the initial state s_0. The policy $\pi(a_t|s_t)$ serves as a mapping from the state space to the distribution of actions. For general RL algorithms, their goal is to search for a policy that maximizes the expected return in Equation (1):

$$\max_{\pi} J_{\rho_0}(\pi, \mathcal{M}) = \max_{\pi} \mathbb{E}_{s_0 \sim \rho_0, \pi} \left[\sum_{t=0}^{H-1} r\left(s_t, a_t\right) \right]. \tag{1}$$

In the model-based offline setting, an approximate MDP model $\hat{\mathcal{M}} = \{\hat{S}, A, \hat{r}, \hat{p}, \rho_0\}$ is learned from a static dataset $D = \{(s_t^i, a_t^i, s_{t+1}^i, r_t^i)\}$, then $\hat{\mathcal{M}}$ is utilized to search for a policy that maximizes the expected return in the model. The transition distribution $\hat{p}$ and the reward function $\hat{r}$ can also be learned from D. Since errors accumulate rapidly when $\hat{\mathcal{M}}$ makes predictions based on its own previous outputs, offline MbRL algorithms may struggle with unstable policy learning. [11] proved that it is crucial to provide a policy with a penalty for visiting the states where the model is likely to be inaccurate. [28] suggested a practical implementation of the reward penalty, i.e., $\tilde{r} = \hat{r} - \lambda u(\hat{s}_t, a_t)$, in which $\hat{r}$ is the model reward. The penalty coefficient λ serves as a hyperparameter chosen for different tasks, and $u(\hat{s}_t, a_t)$ denotes the estimation of the model uncertainty at the state-action tuple $(\hat{s}_t, a_t)$. As such, the objective function is given as below:

$$\max_{\pi} \tilde{J}_{\rho_0}(\pi, \hat{\mathcal{M}}) = \max_{\pi} \mathbb{E}_{s_0 \sim \rho_0, \pi} \left[\sum_{t=0}^{H-1} (\hat{r}_t - \lambda u) \right]. \tag{2}$$

3.2 Evolution Strategy

It is challenging to compute accurate gradients for black-box or noisy optimization problems. Hence, as a derivative-free optimization approach, evolution strategy (ES) has seen a recent revival in the RL community. Instead of optimizing an objective function $F(x)$ directly, ES optimizes the Gaussian smoothing of F: $F_\sigma(x) = \mathbb{E}_{\varepsilon \sim \mathcal{N}(0,I)}[F(x + \sigma\varepsilon)]$, where σ plays the role of a smoothing parameter. For $\sigma > 0$, the function $F_\sigma(x)$ is consistently differentiable, and its gradient is given by $\nabla F_\sigma(x) = (2\pi)^{-d/2} \int_{\mathbb{R}^d} F(x + \sigma\varepsilon) e^{-\frac{1}{2}\|\varepsilon\|_2^2} \varepsilon \mathrm{d}\varepsilon$. Although the gradient is intractable, it can be estimated by a standard Monte Carlo method: $\widehat{\nabla}_N^{\mathrm{s}} F_\sigma(x) = \frac{1}{N\sigma} \sum_{i=1}^N F\left(x + \sigma\varepsilon_i\right) \varepsilon_i$. The estimator is often revised to achieve unbiased estimation with reduced variance. [17] proposed an antithetic

Algorithm 1 Bi-objective policy optimization for model-based offline RL

```
1: input: a S-MDP model with two objectives: the model return $J^{\hat{r}}_{\pi}$ and model uncertainty $J^{q}_{\pi}$.
2: Learn S-MDP on a static dataset.
3: Run an adaptive policy optimization algorithm on S-MDP until convergence:
       $\nabla_{\theta^\pi} \boldsymbol{J}_{\rho_0} = \alpha \nabla_{\theta^\pi} J^{\hat{r}}_{\rho_0} + (1-\alpha) \nabla_{\theta^\pi} J^{q}_{\rho_0}$
   Note that $\alpha$ exists theoretically optimal value according to Equation (4).
```

estimator: $\widehat{\nabla}^{\mathrm{at}}_{N} F_\sigma(x) = \frac{1}{2N\sigma} \sum_{i=1}^{N} \left(F\left(x+\sigma\varepsilon_i\right) - F\left(x-\sigma\varepsilon_i\right)\right)\varepsilon_i$, where the gradient is estimated by the symmetric difference between a perturbation $\varepsilon_i \sim \mathcal{N}(\mathbf{0}, I)$ and its antithetic counterpart $-\varepsilon_i$. Once the ES gradient is obtained, it can be equipped with popular SGD algorithms. A simple way of employing ES to optimize RL policy parameters θ is to set $F(\theta) = J_{\rho_0}(\pi_\theta, \mathcal{M}) \approx \sum_{t=0}^{H-1} r\left(s_t, a_t\right)$. Despite its simplicity, ES achieves competitive performance compared to policy gradient algorithms [4].

4 The Proposed BiES Algorithm

In this section, we first introduce a bi-objective policy optimization framework and illustrate how our method achieves an adaptive trade-off between the model return and model uncertainty. Then we propose the bi-objective evolution strategy (BiES), an efficient and stable policy optimization algorithm, by integrating ES with the framework.

Our framework consists of two steps, as presented in Algorithm 1. First, we propose a surrogate MDP (S-MDP) model and learn it on a static dataset in a supervised manner. Second, based on S-MDP, we develop a provably efficient mechanism to adjust weights between two objectives. The first objective aims to maximize the model return, and the second objective synchronously calibrates the learning bias of the policy. As a result, our approach achieves more stable policy optimization via an adaptive trade-off. More specifically, we construct the S-MDP model $\hat{\mathcal{M}} = \{\hat{S}, A, \hat{r}, q, \hat{p}, \rho_0\}$ by adding $q(\hat{s}, a)$, a guided objective for model uncertainty. We model S-MDP using a bootstrap ensemble of Gaussian dynamics models $\{\hat{\mathcal{P}}^1_\phi, \ldots, \hat{\mathcal{P}}^N_\phi\}$, in which each model of the ensemble is a multi-layer neural network parametrized by ϕ. It predicts the mean μ and covariance Σ of a Gaussian distribution over the next state and reward, $\hat{\mathcal{P}}^i_\phi(\hat{s}^i_{t+1}, \hat{r}^i | \hat{s}_t, a_t) := \mathcal{N}(\mu^i_\phi(\hat{s}_t, a_t), \Sigma^i_\phi(\hat{s}_t, a_t))$. We compute the mean of the rewards predicted by ensemble models as $\hat{r}$, and the minus ensemble difference $-\max_{i,j}(\|\mu^i_\phi(\hat{s}_t, a_t) - \mu^j_\phi(\hat{s}_t, a_t)\|^2)$ as q. Therefore, a bi-objective policy optimization problem can be formulated as:

$$
\begin{aligned}
\max_{\pi} \boldsymbol{J}_{\rho_0}(\pi, \hat{\mathcal{M}}) &= \max_{\pi} (J^{\hat{r}}_{\rho_0}(\pi, \hat{\mathcal{M}}), J^{q}_{\rho_0}(\pi, \hat{\mathcal{M}})), \\
J^{\hat{r}}_{\rho_0}(s, \hat{\mathcal{M}}) &= \mathbb{E}_{s_0 \sim \rho_0, \pi}\left[\sum_{t=0}^{H-1} \hat{r}\left(\hat{s}_t, a_t\right)\right], \\
J^{q}_{\rho_0}(s, \hat{\mathcal{M}}) &= \mathbb{E}_{s_0 \sim \rho_0, \pi}\left[\sum_{t=0}^{H-1} q\left(\hat{s}_t, a_t\right)\right]. \qquad (3)
\end{aligned}
$$

Next, we discuss a special gradient ascent algorithm for solving the bi-objective optimization problem. For example, a general multi-objective optimization problem can be formulated as: $\max_{x\in\mathbb{R}^d} \boldsymbol{f}(x) = \max_{x\in\mathbb{R}^d}(f_1(x), f_2(x), ..., f_m(x))$, where x is a parameter vector; d represents the number of parameters; $\boldsymbol{f}(x)$ is a multi-objective function involving the m sub-objectives $f_{i=1,...,m}$. When we utilize gradient-based methods to solve the problem, the gradient of $\boldsymbol{f}(x)$ is given as follows: $\nabla_x \boldsymbol{f}(x) = \sum_{i=1}^{m} \alpha_i \nabla_x f_i(x)$, s.t. $\sum_{i=1}^{m} \alpha_i = 1, \alpha_i \geq 0$, where we achieve adaptive trade-offs between two even more objectives by automatically tuning the weight $\boldsymbol{\alpha}$. [7] extended the vanilla gradient descent algorithm to multi-objective optimization. His work provides a general method to calculate "optimal" α_i at each gradient update. He proved that the weight $\boldsymbol{\alpha}$ is the optimal solution of a quadratic optimization problem. In the case of a common bi-objective optimization problem, an analytical solution exists as below:

$$\begin{aligned} &\min_{\alpha\in[0,1]} \left\|\alpha\nabla_x f_1(x) + (1-\alpha)\nabla_x f_2(x)\right\|^2 \\ &\alpha = \frac{\left(\nabla_x f_2(x) - \nabla_x f_1(x)\right)^{\mathsf{T}} \nabla_x f_2(x)}{\left\|\nabla_x f_1(x) - \nabla_x f_2(x)\right\|^2} \end{aligned} \tag{4}$$

Now moving back to Equation (3), if we obtain the gradients of $J^{\hat{r}}$ and J^q, the our proposed optimization problem can be solved by a simple gradient method.

$$\theta_{t+1}^{\pi} = \theta_t^{\pi} + \gamma \nabla_{\theta^\pi} \boldsymbol{J}_{\rho_0}, \tag{5}$$

$$\nabla_{\theta^\pi} \boldsymbol{J}_{\rho_0} = \alpha \nabla_{\theta^\pi} J_{\rho_0}^{\hat{r}} + (1-\alpha)\nabla_{\theta^\pi} J_{\rho_0}^{q}, \tag{6}$$

$$\alpha = \frac{\left(\nabla_{\theta^\pi} J_{\rho_0}^{q} - \nabla_{\theta^\pi} J_{\rho_0}^{\hat{r}}\right)^{\mathsf{T}} \nabla_{\theta^\pi} J_{\rho_0}^{q}}{\left\|\nabla_{\theta^\pi} J_{\rho_0}^{\hat{r}} - \nabla_{\theta^\pi} J_{\rho_0}^{q}\right\|^2}. \tag{7}$$

The two sub-objectives $J^{\hat{r}}$ and J^q are both uncertain and noisy functions. With that in mind, ES might be a better choice. We use the antithetic ES estimator to compute their gradients, as listed in Algorithm 2. For better performance, we propose several key improvements. First, any bi-objective optimization algorithm is likely to be stuck prematurely at a bad Pareto stationary point [7], causing no gain in the eventual policy quality. To solve this issue, BiES adopts behavior cloning initialization, which provides a relatively stable policy π_0 as the initial solution by end-to-end behavior cloning. We can intuitively understand that the method works as a regularization of distributional shift. At the initial stage of policy optimization, behavior cloning constrains the policy to the support of training data. Second, we notice that the states of the high dimensional complex tasks take the values in a broad range, which may cause the policies only pay attention to particular features of these states. Therefore, the state normalization can make the policies more robust for multiple-scale state observations: $a_t = \pi(\mathrm{diag}(\Sigma)^{-1/2}(\hat{s}_t - \mu))$. In the offline RL setting, a large-scale dataset is available for the learning algorithms, which means that we can set μ and Σ to be the mean and covariance of all the states in the dataset, just like the data normalization in supervised learning. Finally, BiES adopts an elite selection strategy that sorts noises ε_k in a descending order according to $\max\{J^{\hat{r}}(\pi_{i,k,+}), J^{\hat{r}}(\pi_{i,k,-})\}$ and $\max\{J^{q}(\pi_{i,k,+}), J^{q}(\pi_{i,k,-})\}$, respectively. We only

Algorithm 2 BiES for model-based offline RL

```
1: hyperparameters: step-size $\gamma$, number of noises $N$, smoothing parameter $\sigma$, $b$ elites.
2: inputs: a neural network policy $\pi_i$ parametrized by $\theta_i \in \mathbb{R}^d$. $\theta_0$ is initialized by end-to-end
   behavior cloning, set $\mu$, $\Sigma$ to be the mean and covariance of all states in the dataset $D_{env}$, an
   initial states dataset $\rho_0 \subset D_{env}$, and $i = 0$.
3: train S-MDP $\hat{\mathcal{M}}$ on $D_{env}$.
4: while ending condition not satisfied do
5:     Sample $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_N$ in $\mathbb{R}^d$ from $\mathcal{N}(\mathbf{0}, I)$.
6:     Collect $2N$ rollouts $\{(\hat{s}_t, a_t, \hat{r}_t, q_t)|\pi_{i,k,\pm}\}_{t=0}^{H-1}$ via $\hat{\mathcal{M}}$, where $\pi_{i,k,\pm}$ uses normalized
       states as inputs, and initial state $s_0 \sim \rho_0$.
       $\pi_{i,k,\pm} = \theta_i \pm \sigma\varepsilon_k, k \in \boldsymbol{K} = \{1, 2, ..., N\}$.
7:     Compute objective functions by:
       $J^{\hat{r}} \approx \sum_{t=0}^{H-1} \hat{r}_t, J^q \approx \frac{1}{H} \sum_{t=0}^{H-1} q_t$
8:     Sort the noises $\varepsilon_k$ for $m = \hat{r}, q$:
       $\boldsymbol{K}^m = \mathrm{sort}(\boldsymbol{K}, \max\{J^m_{i,k,+}, J^m_{i,k,-}\})$
9:     Compute the ES gradient of $J^m$ for $m = \hat{r}, q$:
       $g_i^m = \frac{1}{2b\sigma} \sum_{k \in \boldsymbol{K}^m[0:b]} \left(J^m_{i,k,+} - J^m_{i,k,-}\right) \varepsilon_k$
10:    Make the update step:
       $\theta_{i+1} = \theta_i + \text{optimizer-step}(g_i, \gamma)$,
       where $g_i = \alpha g_i^{\hat{r}} + (1-\alpha) g_i^q$, $\alpha$ can be computed by Equation (7)
11:    $i \leftarrow i + 1$
12: end while
```

choose the top-b noises for computing the gradients of two objectives. This enhancement improves the performance of BiES because it reduces the variance of gradient estimation by using more concentrated Gaussian noises.

It is proved that the optimal convex combination of $\nabla f_i(x^*)$ is equal to zero when x^* is Pareto stationary [7] (see Definition 1). However, we use the ES-based gradient estimation, which means $\mathbb{E}(\widehat{\nabla}_{ES} f_i(x^*)) = \nabla f_i(x^*)$. As such, we need to establish an upper bound for the norm of BiES-based gradient $\widehat{\nabla}_{ES} \boldsymbol{f}(x^*)$ when the optimal policy is Pareto stationary. Considering a general multi-objective optimization problem, the necessary conditions for a solution to be optimal are the KKT conditions. Thus each solution that satisfies these conditions is Pareto stationary [18].

Definition 1. *Let x^* be a Pareto stationary solution. Therefore, there exists non-negative scalars $\alpha_1, \ldots, \alpha_m \geq 0$ such that $\nabla \boldsymbol{f}(x^*) = \sum_{i=1}^m \alpha_i \nabla f_i(x^*) = 0, \sum_{i=1}^m \alpha_i = 1$.*

Hence, we can define an ascent direction based on the ES gradient $\widehat{\nabla}_{ES} f_i(x)$, i.e., $\widehat{\nabla}_{ES} \boldsymbol{f}(x) = \sum_{i=1}^m \alpha_i \widehat{\nabla}_{ES} f_i(x)$, s.t. $\sum_{i=1}^m \alpha_i = 1, \alpha_i \geq 0$. Suppose that we have $\widehat{\nabla}_{ES} f_i(x) = \nabla f_i(x) + \epsilon_i$, then $\|\widehat{\nabla}_{ES} f_i(x) - \nabla f_i(x)\| = \|\epsilon_i\| \leq \tilde{\epsilon}_i$. We can prove an upper bound for the norm of $\widehat{\nabla}_{ES} \boldsymbol{f}(x^*)$.

Theorem 1. *If x^* is a solution satisfying Definition 1, then $\|\widehat{\nabla}_{ES} \boldsymbol{f}(x^*)\| \leq \sum_{i=1}^m \alpha_i \tilde{\epsilon}_i$.*

Proof. Since x^* satisfies Definition 1., we have $\nabla \boldsymbol{f}(x^*) = 0$. Consequently,

$$\|\widehat{\nabla}_{ES} \boldsymbol{f}(x^*)\| = \|\sum_{i=1}^m \alpha_i \epsilon_i\| \leq \sum_{i=1}^m \alpha_i \tilde{\epsilon}_i.$$

5 Experiments

In this section, there are three important questions for conducting our experiments:

Q1 Does BiES outperform other SoTA approaches on modern benchmarks?
Q2 Is our proposed BiES essential? Does a single-objective ES perform well?
Q3 How does each component affect the performance of BiES?

Table 1: **Hyper-parameters for BiES**

Parameters	γ	σ	N	b	policy structure
Value	0.02	0.03	30	20	MLP(32,Tanh,32)

Note that we use the same hyper-parameters for all tasks and random seeds. In contrast, prior works, like MOPO [28] and MOReL [13], tune the hyperparameters separately for each benchmark problem. For **Q1**, we pick several offline RL algorithms as baselines, including the model-based and model-free approaches: Model-based policy optimization (MBPO) [11], model-based offline policy optimization (MOPO) [28], behavior regularized actor critic (BRAC) [25], bootstrapping error accumulation reduction (BEAR) [14], and batch-constrained Q-learning (BCQ) [9]. The detailed experimental results are given in Section 5.2. We do not pick MOReL [13] as the baseline because the author-provided implementation of MOReL achieves a lower result than their reported results[3]. Unlike MOReL, [28] argued that MOPO allows the policy to take a few risky actions due to using a soft reward penalty, leading to better exploration. For **Q2**, we compare BiES with a single-objective ES (denoted by **ES w/ p**), and its objective function is given in Equation (2). Moreover, we also evaluate a state-of-the-art multi-objective optimization approach, COMO-CMA-ES [24]. For **Q3**, we conduct a thorough ablation study.

5.1 D4RL Benchmark

D4RL is a standard benchmark for evaluating offline RL algorithms [8]. It provides a variety of environments, tasks, and corresponding datasets containing samples of multiple trajectories $\{(s_i, a_i, r_i, T_i)|\pi_{bc}\}$, where T is the termination flag. We choose three MuJoCo environments (halfcheetah, hopper, walker2d) with five dataset types (random, medium, medium-replay, medium-expert, mixed) as the testbed. **Random** contains 1M samples from a random policy. **Medium** contains 1M samples from a policy trained to approximately 1/3 of the performance of the expert. **Medium-replay** contains the replay buffer of a policy trained up to the performance of the medium agent. **Medium-expert** contains a 50-50 split of medium and expert data (2M samples). **Mixed** is an aggregate of random, medium, and expert datasets (3M samples).

5.2 Experimental Results

To answer **Q1**, the experimental results are given in Table 2. BiES obtains the best mean score over 12 benchmark problems. Among the model-based methods, BiES achieves SoTA results in six out of the 12 problems. In particular, BiES is the strongest by a significant margin on the hopper medium and medium-replay datasets. Meanwhile,

[3] https://github.com/aravindr93/mjrl/issues/35

Table 2: **Experimental results for the D4RL benchmark.** Each number is the normalized score $= \frac{\text{score}-\text{random score}}{\text{expert score}-\text{random score}} \times 100$ of the policy at the last iteration of training (10^6 time steps in total), ± standard deviation, k the length of model rollouts, **CMA-ES** denotes COMO-CMA-ES. We use the results reported by prior works.

Dataset	Environment	BiES (Our)		MOPO		CMA-ES	MBPO	BEAR	BRAC-v	BCQ
		k=1-1000	k=1-500	k=1-5	k=1-500					
random	halfcheetah	**35.7** ± 6.62	24.1 ± 12.3	35.4 ± 2.5	28.2 ± 18.0	2.25 ± 0.01	30.7	25.1	31.2	2.2
	hopper	11.7 ± 1.96	**11.9** ± 0.43	11.3 ± 0.52	11.4 ± 0.47	11.6 ± 0.22	4.5	11.4	12.2	10.6
	walker2d	8.43 ± 7.49	8.91 ± 6.30	**13.6** ± 2.6	4.75 ± 1.74	5.81 ± 0.29	8.6	7.3	1.9	4.9
medium	halfcheetah	**43.0** ± 2.63	41.2 ± 1.39	42.3 ± 1.6	2.79 ± 1.07	42.7 ± 0.79	28.3	41.7	46.3	40.7
	hopper	**90.6** ± 11.9	85.7 ± 13.5	28.0 ± 12.4	48.7 ± 11.1	76.3 ± 22.2	4.9	52.1	31.1	54.5
	walker2d	21.0 ± 12.0	20.1 ± 15.2	17.8 ± 19.3	-0.13 ± 0.01	**42.9** ± 19.5	12.7	59.1	81.1	53.1
med-replay	halfcheetah	32.1 ± 2.27	30.7 ± 1.96	**53.1** ± 2.0	33.6 ± 7.73	32.4 ± 5.97	47.3	38.6	47.7	38.2
	hopper	**93.8** ± 2.95	93.1 ± 6.92	67.5 ± 24.7	62.3 ± 26.7	92.1 ± 2.42	49.8	33.7	0.6	33.1
	walker2d	25.2 ± 14.4	27.1 ± 17.0	**39.0** ± 9.6	17.6 ± 7.11	30.2 ± 9.62	22.2	19.2	0.9	15.0
med-expert	halfcheetah	38.0 ± 4.30	39.2 ± 2.18	**63.3** ± 38.0	0.09 ± 0.51	42.9 ± 1.71	9.7	53.4	41.9	64.7
	hopper	**93.5** ± 11.3	92.8 ± 16.6	23.7 ± 6.0	72.8 ± 20.4	61.8 ± 18.0	56.0	96.3	0.8	110.9
	walker2d	20.7 ± 18.9	15.1 ± 20.6	**44.6** ± 12.9	2.64 ± 4.02	39.6 ± 23.9	7.6	40.1	81.6	57.5
Total Mean		**42.8**	40.8	36.6	23.7	40.1	23.5	39.8	31.4	40.45

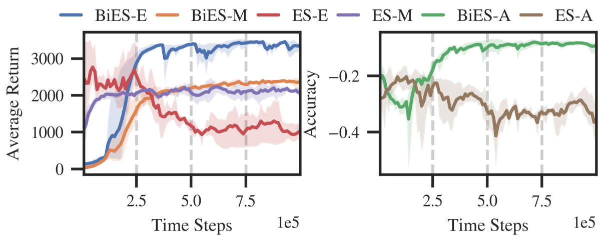

Fig. 2: **Ablation study.** BiES and ES w/ p learning curves in the hopper environment. E: The real MDP return. M: The predicted return. A: The prediction accuracy. ES denotes ES w/ p. Our BiES achieves stable and near-monotonic learning.

COMO-CMA-ES also achieves good results on the walker2d datasets. We hypothesize that the adaptive mechanism of COMO-CMA-ES rapidly decays the step-size when the policy reaches a near-optimal solution, preventing inaccurate update directions from degenerating the learned policy on the walker2d datasets. Such results indicate that the model-based offline policy optimization can benefit from early stopping. Moreover, we find that the larger state-action space in the walker2d environments makes it more difficult to learn a well-generalized model. Fortunately, the recent work shows a powerful γ-model that learns a more accurate state transition for MbRL [12]. Our BiES can attain better performance by being combined with stronger models. It should be pointed out that MOPO utilizes a technique named branched rollout to collect experience replay. A policy begins a rollout from the state s sampled from a static dataset and executes k steps under the learned model. Conversely, BiES learns from the result of the whole rollout $(s_0, s_1, \ldots, s_k)$. Although learning from the whole rollout may damage performance due to accumulated model errors, it will be advantageous in some scenarios where learning algorithms might not directly access the datasets. In Table 2, we compare the performance

Table 3: **Ablation study.** A comparison between BiES and a single-objective ES. **ES w/ p** denotes a vanilla ES algorithm that adopts the reward penalty in Equation (2) (average of five random seeds).

Environment	Dataset	BiES	ES w/ p
hopper	random	11.5 ± 1.96	**12.5** ± 0.63
	medium	**90.6** ± 11.9	33.9 ± 16.5
	med-replay	**93.8** ± 2.95	73.0 ± 20.3
	med-expert	**93.5** ± 11.3	88.2 ± 16.4
	mixed	**72.5** ± 16.5	47.6 ± 16.9
walker2d	random	**8.43** ± 7.49	-0.17 ± 0.06
	medium	**21.0** ± 12.0	-0.11 ± 0.01
	med-replay	**25.2** ± 14.4	0.21 ± 0.64
	med-expert	**20.7** ± 18.9	-0.14 ± 0.09
	mixed	**35.5** ± 22.3	-0.07 ± 0.15

Table 4: **Ablation study.** The effectiveness of behavior cloning initialization (average of five random seeds).

Environment	Dataset	BiES w/ bc	BiES w/o bc
halfcheetah	random	**35.7** ± 6.62	35.0 ± 5.64
	medium	**43.0** ± 2.63	6.39 ± 8.86
	med-replay	**32.1** ± 2.27	28.1 ± 8.62
	med-expert	**38.0** ± 4.30	11.7 ± 17.0
	mixed	**42.9** ± 0.72	41.6 ± 7.23
hopper	random	**11.5** ± 1.96	10.5 ± 0.72
	medium	**90.6** ± 11.9	22.1 ± 32.8
	med-replay	**93.8** ± 2.95	70.7 ± 31.1
	med-expert	**93.5** ± 11.3	92.8 ± 19.7
	mixed	72.5 ± 16.5	**84.1** ± 17.8
walker2d	random	**8.43** ± 7.49	4.73 ± 1.76
	medium	**21.0** ± 12.0	12.2 ± 7.78
	med-replay	**25.2** ± 14.4	14.7 ± 9.27
	med-expert	**20.7** ± 18.9	5.49 ± 3.43
	mixed	**35.5** ± 22.3	6.52 ± 5.81

of BiES and MOPO based on different k steps. When MOPO adopts longer rollouts ($k = 1 - 500$), it performs worse on the walker2d datasets. To answer **Q2**, we pick two complex environments, hopper and walker2d, in which the policy must overcome the severe model exploitation issue. We compare BiES with a single-objective ES (**ES w/ p**). The results are shown in Table 3. It is clear that BiES significantly outperforms **ES w/ p**. Figure 2 records the two methods' learning curves, showing that the superiority of BiES benefits from a better trade-off between the model return and uncertainty estimation. Such results confirm the effectiveness of our method again. To answer **Q3**, we investigate the impact of behavior cloning initialization (**bc**) by comparing the performance of two methods: **BiES w/ bc** and **BiES w/o bc**. In Table 4, we observe apparent performance degradation due to the absence of **bc**. According to [28], it is more challenging for model-based algorithms to learn a well-generalized model from the medium datasets due to the lack of action diversity. However, **BiES w/ bc** obtains significant improvements on these datasets, which reflects the importance of **bc**.

6 Conclusion

This paper proposes a novel approach to address the model exploitation issue in model-based offline reinforcement learning. In contrast to adding a penalty term and user-chosen weight, we propose a bi-objective policy optimization framework where the first objective aims to maximize the model return, and the second one synchronously calibrates the learning bias of the policy. Then we integrate evolution strategy with the framework and develop BiES, an adaptive model-based offline policy optimization algorithm. Experimental results show that our approach achieves state-of-the-art performance compared to other offline RL algorithms.

References

1. Berkenkamp, F., Turchetta, M., Schoellig, A., Krause, A.: Safe model-based reinforcement learning with stability guarantees. In: NeurIPS. pp. 908–918 (2017)
2. Boney, R., Kannala, J., Ilin, A.: Regularizing model-based planning with energy-based models. In: CoRL (2019)
3. Cheng, R., He, C., Jin, Y., Yao, X.: Model-based evolutionary algorithms: a short survey. Complex & Intelligent Systems **4**(4), 283–292 (2018)
4. Choromanski, K., Pacchiano, A., Parker-Holder, J., Tang, Y., Jain, D., Yang, Y., Iscen, A., Hsu, J., Sindhwani, V.: Provably robust blackbox optimization for reinforcement learning. In: CoRL. pp. 683–696 (2020)
5. Chua, K., Calandra, R., McAllister, R., Levine, S.: Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In: NeurIPS (2018)
6. Clavera, I., Rothfuss, J., Schulman, J., Fujita, Y., Asfour, T., Abbeel, P.: Model-based reinforcement learning via meta-policy optimization. In: CoRL (2018)
7. Désidéri, J.A.: Multiple-gradient descent algorithm (mgda) for multiobjective optimization. Comptes Rendus Mathematique **350**(5), 313 – 318 (2012)
8. Fu, J., Kumar, A., Nachum, O., Tucker, G., Levine, S.: D4rl: Datasets for deep data-driven reinforcement learning. In: arXiv,2004.07219 (2020)
9. Fujimoto, S., Meger, D., Precup, D.: Off-policy deep reinforcement learning without exploration. In: ICML (2019)
10. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: ICML. pp. 1861–1870 (2018)
11. Janner, M., Fu, J., Zhang, M., Levine, S.: When to trust your model: Model-based policy optimization. In: NeurIPS (2019)
12. Janner, M., Mordatch, I., Levine, S.: γ-models: Generative temporal difference learning for infinite-horizon prediction. In: arXiv,2010.14496 (2020)
13. Kidambi, R., Rajeswaran, A., Netrapalli, P., Joachims, T.: MOReL : Model-based offline reinforcement learning. arXiv:2005.05951 (2020)
14. Kumar, A., Fu, J., Tucker, G., Levine, S.: Stabilizing off-policy q-learning via bootstrapping error reduction. In: NeurIPS (2019)
15. Levine, S., Kumar, A., Tucker, G., Fu, J.: Offline reinforcement learning: Tutorial, review, and perspectives on open problems. In: arXiv,2005.01643 (2020)
16. Luo, J., Chen, L., Li, X., Zhang, Q.: Novel multitask conditional neural-network surrogate models for expensive optimization. IEEE Trans on Cyber. pp. 1–14 (2020)
17. Mania, H., Guy, A., Recht, B.: Simple random search of static linear policies is competitive for reinforcement learning. In: NeurIPS (2018)
18. Milojkovic, N., Antognini, D., Bergamin, G., Faltings, B., Musat, C.: Multi-gradient descent for multi-objective recommender systems. In: AAAI (2020)
19. Rajeswaran, A., Mordatch, I., Kumar, V.: A game theoretic framework for model based reinforcement learning. In: ICML. pp. 7953–7963 (2020)
20. Salimans, T., Ho, J., Chen, X., Sidor, S., Sutskever, I.: Evolution strategies as a scalable alternative to reinforcement learning. In: arXiv,1703.03864 (2017)
21. Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P.: High-dimensional continuous control using generalized advantage estimation. In: ICLR (2016)
22. Shin, M., Kim, J.: Randomized adversarial imitation learning for autonomous driving. In: IJCAI. pp. 4590–4596 (2019)
23. Sutton, R.S.: Dyna, an integrated architecture for learning, planning, and reacting. ACM Sigart Bulletin (1991)

24. Touré, C., Hansen, N., Auger, A., Brockhoff, D.: Uncrowded hypervolume improvement: Como-cma-es and the sofomore framework. In: GECCO. p. 638–646 (2019)
25. Wu, Y., Tucker, G., Nachum, O.: Behavior regularized offline reinforcement learning. In: arXiv 1911.11361 (2019)
26. Xu, Y., Liu, M., Lin, Q., Yang, T.: Admm without a fixed penalty parameter: Faster convergence with new adaptive penalization. In: NeurIPS. pp. 1267–1277 (2017)
27. Yu, C., Ren, G., Liu, J.: Deep inverse reinforcement learning for sepsis treatment. In: ICHI. pp. 1–3 (2019). https://doi.org/10.1109/ICHI.2019.8904645
28. Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J., Levine, S., Finn, C., Ma, T.: MOPO: Model-based offline policy optimization. arXiv:2005.13239 (2020)