Proceedings of Machine Learning Research vol vvv:1–25, 2024

An Invariant Information Geometric Method for High-Dimensional Online Optimization

Zhengfei Zhang ZF-ZHANG20@MAILS.TSINGHUA.EDU.CN
Yunyue Wei WEIYY20@MAILS.TSINGHUA.EDU.CN
Yanan Sui YSUI@TSINGHUA.EDU.CN
Tsinghua University, Beijing, China

Abstract

Sample efficiency is crucial in optimization, particularly in black-box scenarios characterized by expensive evaluations and zeroth-order feedback. When computing resources are plentiful, Bayesian optimization is often favored over evolution strategies. In this paper, we introduce a full invariance oriented evolution strategies algorithm, derived from its corresponding framework, that effectively rivals the leading Bayesian optimization method in tasks with dimensions at the upper limit of Bayesian capability. Specifically, we first build the framework INVIGO that fully incorporates historical information while retaining the full invariant and computational complexity. We then exemplify INVIGO on multi-dimensional Gaussian, which gives an invariant and scalable optimizer SYNCMA . The theoretical behavior and advantages of our algorithm over other Gaussian-based evolution strategies are further analyzed. Finally, We benchmark SYNCMA against leading algorithms in Bayesian optimization and evolution strategies on various high dimension tasks, including Mujoco locomotion tasks, rover planning task and synthetic functions. In all scenarios, SYNCMA demonstrates great competence, if not dominance, over other algorithms in sample efficiency, showing the underdeveloped potential of property oriented evolution strategies.

Keywords: Invariant optimizer, Information geometry, Evolution strategies, Bayesian optimization

1. Introduction

Without access to gradient information, many real-world continuous-space optimization problems rely on zeroth-order evaluations. These function evaluations are often costly and become less useful over time as the environment changes. Therefore, such tasks are better approached as online optimization problems with zeroth-order feedback and an ignorant initial. An ideal optimizer should have high sample efficiency with reasonable computational complexity.

In this sense, Bayesian optimization (BO) is often the favored choice because it has empirically better sample efficiency in various machine learning scenarios(Frazier, 2018; Shahriari et al., 2015). However, this success is limited to low-dimensional problems due to the cubic computational complexity of its surrogate model(Rasmussen, 2003; Eriksson et al., 2019). Researchers developed versatile scalable variants of Bayesian optimization(Binois and Wycoff, 2022), extending the dominance dimension of Bayesian optimization up to hundreds. In this paper, we use the name of high dimension to denote dimensions range from dozens to hundreds, and the name of online optimization to include both Bayesian optimization and evolution strategies.

Evolution strategies, on the other hand, has a computational complexity that is independent of sample size, which makes it a general method to apply. Over years' development, some theoretical frameworks and guidance are developed(Akimoto et al., 2014, 2012), through which covariance matrix adaptation evolution strategies(CMA-ES) (Hansen, 2016) and its variants(Akimoto and Hansen,

2020; Abdolmaleki et al., 2017) stand out. They are the current leading family of algorithms and reach a balance between sample efficiency and computational cost. However, the lack of a solid theoretical foundation greatly hinders their development despite of many efforts invested(Arnold and Hansen, 2010; Brockhoff et al., 2012; Shirakawa et al., 2018; Nishida and Akimoto, 2018; Ba et al., 2016; Akimoto and Hansen, 2016). The potential of CMA family and even evolution strategies seems to be far from being explored. Instead of developing a solid theory, we want to explore this potential from a property oriented perspective. In specific, we wonder, *can a fully invariant oriented evolution strategies algorithm have great competence against Bayesian optimization in high dimensional tasks?*

The invariant orientation is motivated by general optimization problems where gradient information is available. In this scenario, it is widely known that the performance of leading first-order optimizers such as AdaGrad (Duchi et al., 2011) and Adam (Kingma and Ba, 2014) are highly dependent on the curvature of the optimization objective. Since the curvature depends on the parameterization of the model, parameterization invariant optimizers are thus considered as promising ways. Natural gradient (Amari, 1998) and further efforts (Song et al., 2018; Transtrum and Sethna, 2012) towards practical invariant methods concentrate on exploiting first or higher order structure in parameter space to accelerate or strengthen invariant for optimizers. However, only limited progress has been made. When gradient information is not available, sampling is used in Information Geometric Optimization(IGO)(Ollivier et al., 2017) to estimate the natural gradient for specific parametric distributions. Similarly, geodesic modification is also explored (Bensadon, 2015) but the practical invariant capability is limited as in the general case.

Our contribution. We build the first invariant optimizer framework INVIGO for online optimization with ignorant initial and zeroth-order feedback. Which adopts an approximation to the objective in IGO to allow everywhere differentiability, and a line search strategy to completely and scalably incorporate historical information. When further exemplified with multi-dimensional Gaussian as in CMA family, the derived practical optimizer SYNCMA inherits all properties of INVIGO and has fewer hyperparameters comparing with other CMA optimizers. It is also the first time that historical information is stably incorporated for both mean and covariance parameters. In experiments that benchmark on high dimensional realistic tasks, that Bayesian optimizers usually dominant, and synthetic tasks, SYNCMA demonstrates great competence over other optimizers in sample efficiency.

2. An Invariant Optimizer Family with an Approximate Objective

Considering the online optimization problem where a black-box function f needs to be optimized, and the optimizer is initially ignorant with only zeroth-order feedback available.

$$x^* = \operatorname{argmin}_{x\in\mathbb{R}^n} f(x) \tag{1}$$

A global parametric sample distribution $\theta \mapsto p_\theta$ is often used to relax the original optimization problem into an optimization problem on the parameter space Θ, with a substitutional fitness function $g_{f,\theta}(x)$ to represent how good a sample $x \in \mathbb{R}^n$ is.

$$\theta^* = \operatorname{argmin}_\theta \mathbb{E}_{p_\theta}[g_{f,\theta}(x)] \tag{2}$$

The expression of g is determined manually from the set of integrable functions depending on original objective f and perhaps the current point $\theta \in \Theta$. The substitution here actually generalize

the problem as specifically, (1) if f is good, then it is naturally to set $g_{f,\theta} = f$; (2) if g is related with θ, then the problem will match the time-varying environment setting for which lots of online optimizers pursue.

2.1. Natural Gradient Flow with Zeroth-order Feedback

To solve equation (2) on the parameter space Θ, a natural gradient flow is often the primary choice.

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = -\tilde{\nabla}_\theta \mathbb{E}_{p_\theta}[g_{f,\theta^t}(x)] \tag{3}$$

Here θ^t denotes the current θ with static nature that gradient should not apply on. The usage of natural gradient keep this ODE invariant under smooth bijective transformation of parameter space. Its vanilla discrete version, i.e. natural gradient descend algorithm goes to,

$$\theta^{t+1} = \theta^t - h\tilde{\nabla}_\theta|_{\theta=\theta^t} L_{\theta^t}(\theta) \tag{4}$$

Here h denotes the learning rate, and the loss function is defined as $L_{\theta^t}(\theta) \equiv \mathbb{E}_{p_\theta}[g_{f,\theta^t}(x)]$. This definition of loss function can be easily extended to contain the usual definition of loss function in deep learning by extending the sample distribution p_θ to include neural network, although in this paper we concentrate on parametric distribution families.

To practically compute the natural gradient in black-box setting, IGO set a sampling method for certain distribution families,

$$\tilde{\nabla}_\theta|_{\theta=\theta^t} L_{\theta^t}(\theta) = I^{-1}(\theta^t) \int g_{f,\theta^t}(x) \frac{\partial \ln p_\theta(x)}{\partial \theta}|_{\theta=\theta^t} p_{\theta^t}(\mathrm{d}x) \tag{5}$$

In IGO, the natural gradient is only available at point θ^t, different choices of distribution family Θ thus give different optimization methods in the form of (4). We thus define the IGO complexity to measure the computational complexity of other natural gradient based optimizers.

Assumption 1 *For a given $x \in \mathbb{R}^n$ and $\theta \in \Theta$, $I^{-1}(\theta)\frac{\partial \ln p_\theta(x)}{\partial \theta}$ cost finite $\mathcal{O}(H)$ time to compute.*

Definition 1 (IGO complexity) *When assumption 1 holds, the IGO complexity $\mathcal{O}(HN)$ is the computational complexity for single step updates when applying IGO to natural gradient method, i.e. to compute equation (5) with N samples.*

When discretizing with certain learning rate, errors with respect to the invariant property occur, which may accumulate to drastically change the trajectory. In gradient accessible setting with general lost function, the best invariant error achieved (Song et al., 2018) is 2-nd order invariant, representing the decreasement of the error between the optimizer and some exactly invariant trajectories is $\mathcal{O}(h^2)$. Similar error order is achieved in the content of black-box setting for certain parametric distribution (Bensadon, 2015). There are some attempts to illustrate a fully invariant optimizer, such as IGO-ML in (Ollivier et al., 2017), but they are not practically invented. In short, there is no practical algorithm that is fully invariant or even have a better error order.

Definition 2 (Invariant property) *Let θ be the parameter of an optimizer using model p_θ and $\varphi(\theta)$ be an smooth bijective transformation of θ of the same optimizer using model $p'_{\varphi(\theta)} = p_\theta$. Let θ^t be the optimization trajectory when optimizing objective f, parameterized by θ and initialized at θ^0. And φ^t the optimization trajectory when optimizing objective f, parameterized by φ and initialized at $\varphi^0 = \varphi(\theta^0)$. We say this optimizer is invariant if $\forall t \in \mathbb{N}, \varphi^t = \varphi(\theta^t)$.*

2.2. Optimizing with the Approximate Objective

Given that g_{f,θ^t} is manually selected and $\forall b \in \mathbb{R}, \nabla_\theta E_{p_\theta}[g_{f,\theta^t}(x)] = \nabla_\theta E_{p_\theta}[g_{f,\theta^t}(x) + b]$, we assume g_{f,θ^t} to be non-negative without loss of generality. Let reweighted distribution $q_\theta(x) \equiv \frac{p_\theta(x) g_{f,\theta^t}(x)}{L_{\theta^t}(\theta)}$, then we can decompose $\log L_{\theta^t}(\theta)$ as follow,

$$\log \frac{L_{\theta^t}(\theta)}{L_{\theta^t}(\theta^t)} = D_{KL}(q_{\theta^t}||q_\theta) + D_{KL}(q_{\theta^t}||p_{\theta^t}) - D_{KL}(q_{\theta^t}||p_\theta) \tag{6}$$

Inspired from this decomposition, we claim $D_{KL}(q_{\theta^t}||p_\theta)$ a good objective approximating $L_{\theta^t}(\theta)$. All the proofs and detailed derivations are set in Appendix[1] A.

Proposition 3 *The KL-divergence $D_{KL}(q_{\theta^t}||p_\theta)$ is a substitution for $L_{\theta^t}(\theta)$ with the following properties.*

1. *The (natural) gradients for $\log L_{\theta^t}(\theta)$ and $-D_{KL}(q_{\theta^t}||p_\theta)$ coincide at current point θ^t, further $\forall \theta \equiv (\theta^t + \delta\theta) \in \Theta$, $\nabla_\theta \log L_{\theta^t}(\theta) = -\nabla_\theta D_{KL}(q_{\theta^t}||p_\theta) + O(\delta\theta)$.*
2. *Under Assumption 1 , computing natural gradient of $D_{KL}(q_{\theta^t}||p_\theta)$ at any point $\theta \in \Theta$ costs the IGO complexity $O(HN)$. While objective $L_{\theta^t}(\theta)$ in IGO is only available to be differentiated at point θ^t.*

Combining with above two properties of $D_{KL}(q_{\theta^t}||p_\theta)$, it is natural to consider a step size constraint update for θ^{t+1} when optimizing $D_{KL}(q_{\theta^t}||p_\theta)$. The specific choice of the step size constraint comes from the definition of natural gradient,

$$\tilde{\nabla}|_{\theta=\theta^t} L_{\theta^t}(\theta) \propto \lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \text{argmax}_{\delta\theta \ s.t. \ D_{KL}(p_{\theta^t}||p_{\theta^t+\delta\theta}) \leq \epsilon^2/2} D_{KL}(q_{\theta^t}||p_{\theta^t+\delta\theta}) \tag{7}$$

Now our optimization problem for each time step is,

$$\theta_*^{t+1} = \text{argmax}_\theta D_{KL}(q_{\theta^t}||p_\theta) \ \ s.t. \ D_{KL}(p_{\theta^t}||p_\theta) \leq \epsilon^2/2 \tag{8}$$

It is notable that the optimization problem (8) is approximately solving $\frac{\mathrm{d}\theta}{\mathrm{d}t} = -s(\theta)\tilde{\nabla} \log L_{\theta^t}(\theta)$. Here $s(\theta) \equiv \frac{1}{\epsilon}||\tilde{\nabla}|_{\theta=\theta^t} D_{KL}(q_{\theta^t}||p_\theta)||$ corresponds to the implicit learning rate of INVIGO. As there is no explicit learning rate, this implicit learning rate hints an overall learning rate that is proportion to $D_{KL}(p_{\theta^t}||p_\theta)^{-0.5}$. The efficiency of such dependency of the overall learning rate with the KL-divergence between adjacent distributions is widely verified in natural gradient based optimization methods such as K-FAC (Ba et al., 2016) and reinforcement learning algorithms such as ACKTR (Wu et al., 2017). Our formulation is justified thereby.

2.3. Invariantly Incorporating Historical Information

When only local information is used in each iteration, historical information is useful for the optimization, even if the environment is continuously changing over time. We thus modify objective

1. Please refer to https://github.com/Anoxxx/SynCMA-official for appendix and source code.

$D_{KL}(q_{\theta^t}||p_\theta)$ to incorporate historical information. Here T denotes the horizon and the widely used exponential decay is applied with decay parameter $\lambda \in [0, 1)$, λ^0 is regarded as 1 in default.

$$\theta_*^{t+1} = \text{argmax}_\theta \sum_{\tau=0}^{T} \lambda^\tau D_{KL}(q_{\theta^{t-\tau}}||p_\theta) \;\; s.t.\; D_{KL}(p_{\theta^t}||p_\theta) \leq \epsilon^2/2 \tag{9}$$

Although problem (9) can be solved with strong duality and additional convex optimization, applying a simple natural Lagrange condition will yields more room for accessible invariant with proper computational cost. Here we denote $G^t(\theta) \equiv \sum_{\tau=0}^{T} \lambda^\tau D_{KL}(q_{\theta^{t-\tau}}||p_\theta)$,

$$\tilde{\nabla}_\theta|_{\theta=\theta^{t+1}}(-G^t(\theta) + \eta(\epsilon^2/2 - D_{KL}(p_{\theta^t}||p_\theta))) = 0 \tag{10}$$

We thus name such algorithm family from iteratively solving (10) for different choice of parametric distribution family Θ as INVIGO. Further, when T is set to be large, we can always replace $G^t(\theta)$ with a self-evovled term $M^t(\theta)$ that retain the gradient information. In practice, it suffice to evolve only $\tilde{\nabla}_\theta M^t(\theta)$.

$$\tilde{\nabla}_\theta G^t(\theta) = -\tilde{\nabla}_\theta M^t(\theta) + \tilde{\nabla}_\theta D_{KL}(q_{\theta^t}||p_\theta) \tag{11}$$

Assumption 2 *The chosen fitness function $g_{f,\theta^t}(x)$ and the Lagrange multiplier η are independent from the parameterization of θ.*

Theorem 4 (Invariant for INVIGO) *When assumption 1, 2 hold and the decay weight λ is independent from the parameterization of θ, optimizers in INVIGO are invariant and the single step computational cost is $\mathcal{O}(\min(HNT, HN + K))$ where $\mathcal{O}(K)$ denotes the cost to compute $\tilde{\nabla}_\theta M^t(\theta)$.*

3. Exemplifying with Multi-dimensional Gaussian

We choose multi-dimensional Gaussian as our candidate distribution family Θ given its wide applications. To start with, we claim the computational accessibility of multi-dimensional Gaussian for assumption 1,

Proposition 5 (Theorem 4.1 in Akimoto et al. (2012)) *Suppose θ_m and θ_c are n- and $n(n+1)/2$-dimensional column representing mean and covariance respectively. Then $\partial m/\partial\theta_m$ and $\partial vec(C)/\partial\theta_c$ are invertible at $\theta \in \Theta$ and,*

$$I_m^{-1}(\theta)\frac{\partial \ln P_\theta(x)}{\partial \theta_m} = (\frac{\partial m}{\partial \theta_m})^{-1}(x - m) \tag{12}$$

$$I_c^{-1}(\theta)\frac{\partial \ln P_\theta(x)}{\partial \theta_c} = (\frac{\partial vec(C)}{\partial \theta_c})^{-1} vec((x - m)(x - m)^T - C) \tag{13}$$

Then we propose our choices of the fitness function $g_{f,\theta^t}(x)$ and the Lagrange multiplier η to satisfy assumption 2 while being as simple as possible. We denote the fitness function $g_{f,\theta^t}(x)$ as the level function that reflect the probability to sample a better value from p_{θ^t}. This is the exact choice used in standard CMA-ES Hansen (2016) . In time step t, N samples $\{x_i^t\}$ are drawn from p_{θ^t} and we further denote $\hat{w}_i^t \equiv \frac{g_{f,\theta^t}(x_i^t)}{\sum_i g_{f,\theta^t}(x_i^t)}$ as the normalized fitness for sample x_i^t.

According to proposition 5, parameter $\theta = (\theta_m, \theta_c) \mapsto \mathcal{N}(m, C)$ with $\theta_m \in \mathbb{R}^n$ and $\theta_c \in \mathbb{R}^{n(n+1)/2}$ representing mean and covariance respectively. We can thus split the Lagrange multiplier into $\eta = (\eta_m, \eta_c)$ in INVIGO without violating the invariant property. For better comparisons with CMA family optimizers, we adopt this split to directly use the default values in the fine-tuned version of CMA-ES and keep them constant. The assumption 2 is satisfied therefore.

We use the parameterization $\theta = (m, C)$ for simplification sake through this section. Different parameterizations that meet the conditions in proposition 5 will conduct different practical optimizers by following this section with minor modifications. The performance should be the same up to the transformation due to the invariant property.

3.1. An Invariant Optimizer with Historical Information : SYNCMA

We directly apply INVIGO and a maximum time horizon $T = t - 1$. By choosing such infinite horizon, the historical information is maximally used. To reduce the computational costs to the same as IGO, i.e. scalable to $\mathcal{O}(HN)$ for single step update, we design $M^t(\theta)$ as follow,

$$\begin{aligned}\tilde{\nabla}_m M^t(\theta) &= \lambda_0(s_m^t + m^t - m) \\ \tilde{\nabla}_c M^t(\theta) &= \lambda_0((s_c^t + m^t - m)(s_c^t + m^t - m)^T - C) + Q_1^t + Q_2^t \circ m + Q_3^t m m^T\end{aligned} \tag{14}$$

Here scalars $\lambda_0, Q_1^t \in \mathbb{R}$ and vectors $s_m^t, s_c^t, Q_2^t, Q_3^t \in \mathbb{R}^n$, with $\circ$ is used to denote $v_1 \circ v_2 \equiv v_1 v_2^T + v_2 v_1^T$ for two vectors $v_1, v_2 \in \mathbb{R}^n$. For simplicity needs, we denote $d_i^t \equiv x_i^t - m^t, d_w^t \equiv \sum_i \hat{w}_i^t d_i^t, \hat{d}_w^t \equiv d_w^t + m^t$ to represent statistics in a single generation, and $\hat{s}_m^{t-1} \equiv s_m^{t-1} + m^{t-1}$, $\hat{s}_c^{t-1} \equiv s_c^{t-1} + m^{t-1}$ to represent elements for history. Corresponding updates for hyperparameter $\lambda_0 \in \mathbb{R}$ and self-evolved terms $s_m^t, s_c^t, Q_2^t, Q_3^t \in \mathbb{R}^n$ that initially zero are shown below.

$$\lambda = \lambda_0 / \lambda_0 + 1 \tag{15}$$

$$s_m^t + m^t = \lambda \hat{s}_m^{t-1} + (1 - \lambda)\hat{d}_w^{t-1} \tag{16}$$

$$s_c^t + m^t = \sqrt{\lambda}\hat{s}_c^{t-1} + \sqrt{1 - \lambda}\hat{d}_w^{t-1} \tag{17}$$

$$Q_1^t = \lambda Q_1^{t-1} + \lambda \sum \hat{w}_i (d_i^{t-1} - d_w^{t-1})(d_i^{t-1} - d_w^{t-1})^T - \lambda_0 \sqrt{\lambda}\sqrt{1 - \lambda}\hat{d}_w^{t-1} \circ \hat{s}_c^{t-1} \tag{18}$$

$$Q_2^t = \lambda Q_2^{t-1} - \lambda_0(\sqrt{\lambda} + \sqrt{1 - \lambda} - 2)(\sqrt{\lambda} * \hat{s}_c^{t-1} + \sqrt{1 - \lambda} * \hat{d}_w^{t-1}) \tag{19}$$

$$Q_3^t = \lambda Q_3^{t-1} - \lambda_0(\sqrt{\lambda} - 1)(\sqrt{1 - \lambda} - 1) \tag{20}$$

We now arrive at the single step update for next parameter $\theta^{t+1} = (m^{t+1}, C^{t+1})$. The resulting algorithm is named as SYNCMA to emphasize another prominent characterization, the synchronous update nature, as discussed in section 3.2, besides invariance. The final updates in single iteration with $z_m = \eta_m + \lambda_0 + 1, z_c = \eta_c + \lambda_0 + 1, \beta^t = \frac{1}{z_m}(d_w^t + \lambda_0 s_m^t)$ for brevity sake is shown below.

$$m^{t+1} = m^t + \beta^t \tag{21}$$

$$\begin{aligned}C^{t+1} = &\frac{\eta_c}{z_c}(C^t + \beta^t(\beta^t)^T) + \frac{\lambda_0}{z_c}(s_c^t - \beta^t)(s_c^t - \beta^t)^T \\ &+ \frac{1}{z_c}(\sum_i \hat{w}_i (d_i^t - \beta^t)(d_i^t - \beta^t)^T + Q_1^t + Q_2^t \circ m^t + Q_3^t m^t (m^t)^T)\end{aligned} \tag{22}$$

3.2. Theoretical Comparison with Other CMA Optimizers

Theoretical advantages We here list four main theoretical advantages of SYNCMA over other CMA optimizers : it is invariant, it has less hyperparamters, it fully incorporates historical information, and it is synchronous.

The invariant and historical information fully incorporated properties are grant by induced framework INVIGO . In the previous literature, no optimizers ever incorporate historical information into the mean update with a stable or invariant procedure.

There is only three hyperparamters $\lambda_0, \eta_m, \eta_c$ for SYNCMA, and these hyperparametres all show up in all other optimizers. Besides, other optimizers such as CMA-ES also have additional hyperparamters such as the weights and the initial value of learning rate σ in the parameterization of $\theta = (m, \sigma^2\Sigma)$, and the weights for evolving historical information term.

The synchronous property from which SYNCMA is named as a result of the fact that INVIGO is built on IGO, which treats the current distribution θ as a single point in space to update, i.e. the updates for mean and covariance are intertwined. For other CMA algorithms, the updates are performed sequentially in each generation, e.g. $m^{t+1} = U_m(m^t, (\sigma^t)^2\Sigma^t)$, $\Sigma^{t+1} = U_c(m^{t+1}, (\sigma^t)^2\Sigma^t)$, etc. Moreover, to strictly follow the proposition 5 , updates need to be intertwined as SYNCMA does.

Connection to CMA-ES It is also worth making the connection between SYNCMA and the CMA family algorithms. Where the approximations used correspond to the four theoretical advantages of SYNCMAmentioned above.

Proposition 6 *When (1) the historical information is partially used for covariance, i.e.* $\tilde{\nabla}_m M^t(\theta) = 0$ *and* $\tilde{\nabla}_c M^t(\theta) = \lambda_0((s_c^t + m^t - m)(s_c^t + m^t - m)^T - C)$. *(2) all the higher order terms, when assuming* $\eta_c \approx z_c \gg 1, z_m \gg 1$, *are discarded.* SYNCMA *coincide with CMA-ES up to an external learning rate difference.*

4. Experiments

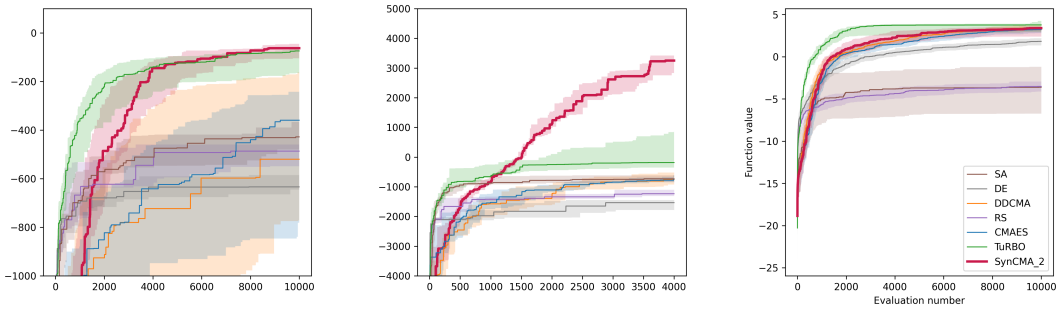
In this section, we evaluate SYNCMA with other baselines in Mujoco locomotion tasks, rover planning task and synthetic functions. The criteria are chosen in the context of online optimization, focusing on full optimization procedures in the natural axis and sample efficiency when achieve a near global value. All optimization procedures are plotted with the shaded area bounded by quantiles and the solid line denoting the median performance over all trails.

Baselines are chosen in a structured way. First, random search (RS) (Bergstra and Bengio, 2012) is chosen as the overall baseline. Then, two black-box optimizers, differential evolution (DE) (Storn and Price, 1997) and simulated annealing (SA) (Bouttier and Gavra, 2019) are chosen. Among the CMA optimizers, we choose CMA-ES and two of its leading variants DD-CMA(Akimoto and Hansen, 2020) and TR-CMA-ES(Abdolmaleki et al., 2017) for detailed comparison. Finally, the Bayesian optimization method TuRBO (Eriksson et al., 2019) is used as the state-of-the-art baseline for BO. Parameters η_m, η_c of SYNCMA are set to constant that match the initial settings of the corresponding parameters in CMA-ES. λ_0 corresponds to a combination of several parameters in CMA-ES so we simply test with the constant value $\lambda_0 = 2$, which corresponds to the approximate counterpart in CMA-ES. We use this value throughout the main paper while there are better performances of SYNCMA with different λ_0 as shown in ablation studies in Appendix B.4. The intial distribution for all Gaussian based optimizers are identity matrix.

TR-CMA-ES is based on its original paper version implemented in Matlab due to precision problems in Python, and therefore we exclude TR-CMA-ES in the Mujoco locomotion and rover planning tasks as they are based on specific Python libraries. All other baselines are implemented with their fine-tuned version available online (Duan et al., 2022; Balandat et al., 2020). See Appendix B for details.

4.1. Mujoco Locomotion Task

We first evaluate SYNCMA and other baselines on the widely tested Mujoco locomotion tasks (Todorov et al., 2012), which are popular benchmarks for Bayesian optimization and reinforcement learning algorithms. To run sampling-based optimizers on Mujoco, we refer to (Mania et al., 2018) and optimize a linear policy: $\mathbf{a} = \mathbf{W}\mathbf{s}$, where $\mathbf{a}$ is the agent action and $\mathbf{s}$ is the environment state. The parameter matrix $\mathbf{W}$ are continuous and in the range of $[-1, 1]$. Among all 6 tasks, we dismiss the overly high dimensional task Humanoid(6392d) and test all other 5 tasks with batch size $N = 100$. Two results are shown here in figure 1(*a*), 1(*b*) with more results in Appendix B.1. While TuRBO dominates other baselines, SYNCMA outperforms TuRBO in 2 tasks and remains competitive with TuRBO for the other 3 tasks.



(*a*) Ant(888d) (*b*) HalfCheetah(102d) (*c*) Rover(60d)

Figure 1: Optimization procedure for two high dimensional Mujoco locomotion tasks over 10 trials and rover planning task over 100 trails. Index of SYNCMA indicate λ_0.

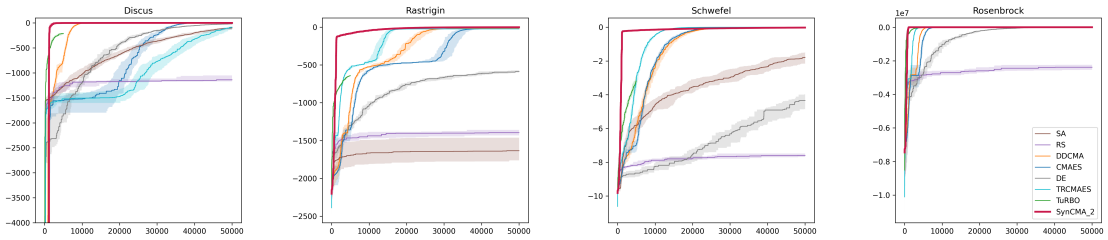
4.2. Rover Planning Task

To further explore the empirical performance of SYNCMA in a realistic setting, we consider the rover trajectory optimization task, where a start position s and a goal position g are defined in the 2D plane, as well as a cost function $c(x)$ over the state space. The trajectories are described by a set of points to which a B-spline is fitted and the cost function is computed. The whole state space is $x \in [0, 1]^{60}$ and we make the batch size $N = 2n = 120$. A reward function to be optimized is defined to be non-smooth, discontinuous, and concave over the first two and last two dimensions of the state. The result in figure 6(*f*) shows that SYNCMA still exhibits competitive performance over other baselines.

4.3. Synthetic Function

We finally select 10 commonly used synthetic functions with dimension n arbitrarily set. This is the more traditional testbed for black-box optimization and specific evolution strategies studies. These functions, including different characteristics such as multi-model, ill-conditioned and ill-scaled, are scaled to a global minimum value 0 with shifted domain. The batch size is $N = 2n$ and the evaluation limit is the same for all optimizers except TuRBO, where the budget is fixed at 5,000 evaluations due to memory limitations.

The full experiments are run with different dimensions of $n = \{32, 64, 128\}$ and the results are presented in two ways under the same evaluation budget: near global optimum performance and the whole optimization procedure. Limited results are presented here and please refer to Appendix B.3 for the full contents.



(*a*) Discus (*b*) Rastrigin (*c*) LevyMontalvo (*d*) Rosenbrock

Figure 2: Optimization procedure in 4 typical synthetic functions with dimension $n = 64$ over 20 trails considering all optimizers. Index of SYNCMA indicate λ_0.

Table 1: Near global optimum performance on 64d synthetic functions(lower is better) over 20 trials with budget of 50000 evaluations, TuRBO is excluded due to memory limitation. Numbers in brackets indicates the median evaluation number needed for optimizers to achieve value better than 0.5.

Optimizer	Sphere	Discus	Schwefel	DiffPowers	LevyMontalvo	Rastrigin	Ackley
SA	0.6	100.7	0.1(2650)	37.5	6.8	1634.7	13.6
RS	793.9	1138.8	29902.6	2369.4	14.4	1395.9	11.8
DE	7.2	20.0	25.6	63.6	0.4(49700)	586.9	3.1
DDCMA	0.0(10047)	**0.0(12748)**	0.0(16820)	0.0(10164)	0.0(9087)	17.9	0.0(11757)
CMAES	0.0(13825)	0.0(43265)	0.0(16134)	0.0(18503)	0.0(9901)	21.4	0.0(15620)
TRCMAES	0.0(7185)	85.9	0.0(9905)	0.0(12040)	0.0(5515)	22.4	0.0(7869)
SYNCMA(**Ours**)	**0.0(3938)**	0.0(18820)	**0.0(1157)**	**0.0(1158)**	**0.0(2318)**	**0.2(42696)**	**0.0(7567)**

According to table 1 where TuRBO is excluded as it is unable to scale to this budget, SYNCMA demonstrate both superior optimization capability and efficiency over others. While other optimizers are less efficient and fail to optimize high-conditional number multi-model function Rastrigin, ill-scaled function Discus and others. Further, full optimization procedures including TuRBO with maximum budget under storage limit are partially shown in figure 2, SYNCMA still outperforms others including TuRBO after first several hundreds evaluation from 32 to 128 dimension, demonstrating the capability of such optimizer derived from an invariant framework.

4.4. Ablation Study

The weight for historical information λ_0 is a parameter that substitutes a combination of several parameters in CMA optimizers, and is set constantly as $\lambda_0 = 2$. We thus study the sensitivity on this parameter for constant setting here. All of previous experiments are repeated for $\lambda_0 \in [0, 4]$, with results for $\lambda_0 = \{0, 1, 2, 4\}$ shown in Appendix B.4, from which we summarize several observations within range $[0, 4]$ here.

Sensitivity. When SYNCMA includes historical information, i.e. $\lambda_0 > 0$, SYNCMA consistently shows competitive performance.

Function Landscape. When their exists a fundamental subspace that covers the structure of the problem, as in Rastrigin, a higher λ_0 yields better performance and efficiency. Otherwise, as in LevyMontalvo, a higher λ_0 might be detrimental.

Dimensionality. Observed from tasks in Mujoco, synthetic functions, and rover planning, a higher dimension generally requires a higher λ_0.

5. Limitations

There is still much to explore in both framework INVIGO and optimizer SYNCMA . For the framework, we use a line search for each step to yield rooms for invariant and complexity, and it may need more theoretical results to characterize its behavior. Moreover, it currently only works for certain parametric distribution families that IGO set. As we are based on the approximation of the IGO objective, it is possible to generalize to a broader family of models such as neural networks. Additionally, the utilized line search hamper a rigorous theoretical analysis, which may of single interest to design per step subroutine while keep the framework invariant and scalable.

For the algorithm, we set parameters η_m, η_c and λ_0 constants in the experiments, which might make SYNCMA overshoot in the final stage of optimization when it is close to the optimum. However, they can absolutely be invariant variables with evolving rules for a better overall optimization performance.

6. Conclusion

We present an invariant optimizer framework INVIGO that fully incorporates historical information. When exemplified with multi-dimensional Gaussian, our framework derives a invariant optimizer SYNCMA that retains the computational complexity as in information geometric optimization. With a straightforward invariant oriented motivation, SYNCMA shows competitive performance in both realistic and synthetic scenarios against leading Bayesian and evolution strategies optimizers.

We highlight its performance on high dimensional realistic problem as it shows the potential of a property oriented evolution strategies optimizer against Bayesian optimization optimizers. And we also defend the importance of fully invariant in optimization. Moreover, we believe the property oriented perspective is more approachable than inventing a rigorous theory that illustrates and improves current methods.

References

Abbas Abdolmaleki, Bob Price, Nuno Lau, Luis Paulo Reis, and Gerhard Neumann. Deriving and improving cma-es with information geometric trust regions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 657–664, 2017.

Youhei Akimoto and Nikolaus Hansen. Online model selection for restricted covariance matrix adaptation. In *International Conference on Parallel Problem Solving from Nature*, pages 3–13. Springer, 2016.

Youhei Akimoto and Nikolaus Hansen. Diagonal acceleration for covariance matrix adaptation evolution strategies. *Evolutionary computation*, 28(3):405–435, 2020.

Youhei Akimoto, Yuichi Nagata, Isao Ono, and Shigenobu Kobayashi. Theoretical foundation for cma-es from information geometry perspective. *Algorithmica*, 64:698–716, 2012.

Youhei Akimoto, Anne Auger, and Nikolaus Hansen. Comparison-based natural gradient optimization in high dimension. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 373–380, 2014.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Dirk V Arnold and Nikolaus Hansen. Active covariance matrix adaptation for the (1+ 1)-cma-es. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 385–392, 2010.

Jimmy Ba, Roger Grosse, and James Martens. Distributed second-order optimization using kronecker-factored approximations. In *International Conference on Learning Representations*, 2016.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL http://arxiv.org/abs/1910.06403.

Jérémy Bensadon. Black-box optimization using geodesics in statistical manifolds. *Entropy*, 17(1): 304–345, 2015.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

Mickael Binois and Nathan Wycoff. A survey on high-dimensional gaussian process modeling with application to bayesian optimization. *ACM Transactions on Evolutionary Learning and Optimization*, 2(2):1–26, 2022.

Clément Bouttier and Ioana Gavra. Convergence rate of a simulated annealing algorithm with noisy observations. *The Journal of Machine Learning Research*, 20(1):127–171, 2019.

Dimo Brockhoff, Anne Auger, and Nikolaus Hansen. On the effect of mirroring in the ipop active cma-es on the noiseless bbob testbed. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, pages 277–284, 2012.

Qiqi Duan, Guochen Zhou, Chang Shao, Zhuowei Wang, Mingyang Feng, Yijun Yang, Qi Zhao, and Yuhui Shi. Pypop7: A pure-python library for population-based black-box optimization. *arXiv preprint arXiv:2212.05652*, 2022.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Kouhei Nishida and Youhei Akimoto. Psa-cma-es: Cma-es with population size adaptation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 865–872, 2018.

Yann Ollivier, Ludovic Arnold, Anne Auger, and Nikolaus Hansen. Information-geometric optimization algorithms: A unifying picture via invariance principles. *The Journal of Machine Learning Research*, 18(1):564–628, 2017.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2015.

Shinichi Shirakawa, Youhei Akimoto, Kazuki Ouchi, and Kouzou Ohara. Sample reuse via importance sampling in information geometric optimization. *arXiv preprint arXiv:1805.12388*, 2018.

Yang Song, Jiaming Song, and Stefano Ermon. Accelerating natural gradient with higher-order invariance. In *International Conference on Machine Learning*, pages 4713–4722. PMLR, 2018.

Rainer Storn and Kenneth Price. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341, 1997.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

Mark K Transtrum and James P Sethna. Geodesic acceleration and the small-curvature approximation for nonlinear least squares. *arXiv preprint arXiv:1207.4999*, 2012.

Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *Advances in neural information processing systems*, 30, 2017.

Appendices : An Invariant Information Geometric Method for High-dimensional Online Optimization

Appendix A. Proofs and Derivations

A.1. Proof for Theorem 3

Proof To prove the former part, it is suffice to notice,

$$\nabla_\theta \log L_{\theta^t}(\theta) - (-\nabla_\theta D_{KL}(q_{\theta^t}||p_\theta)) = \nabla_\theta D_{KL}(q_{\theta^t}||q_\theta) \tag{23}$$

$$= \nabla_\theta(\frac{1}{2}\sum I_{ij}^{q_{\theta^t}}(\theta^t)\delta\theta_i\delta\theta_j + O(\delta\theta^3)) \tag{24}$$

To prove the latter part, we first apply natural gradient to $L_{\theta^t}(\theta)$,

$$\tilde{\nabla}_\theta L_{\theta^t}(\theta) = \tilde{\nabla}_\theta \int g_{f,\theta^t}(x)p_\theta(x)\mathrm{d}x \tag{25}$$

$$= I^{-1}(\theta)\int g_{f,\theta^t}(x)\frac{\partial \ln p_\theta(x)}{\partial\theta}\frac{p_\theta(x)}{p_{\theta^t}(x)}p_{\theta^t}(\mathrm{d}x) \tag{26}$$

In contrast, applying natural gradient to $D_{KL}(q_{\theta^t}||p_\theta)$,

$$\tilde{\nabla}_\theta D_{KL}(q_{\theta^t}||p_\theta) = -I^{-1}(\theta)\int \frac{g_{f,\theta^t}(x)}{E_{p_{\theta^t}}[g_{f,\theta^t}(x)]}\frac{\partial \ln p_\theta(x)}{\partial\theta}p_{\theta^t}(\mathrm{d}x) \tag{27}$$

Thus under assumption 1 , only $D_{KL}(q_{\theta^t}||p_\theta)$ is available to take natural gradient on any point while the computational cost remain the same ■

A.2. Proof for Theorem 4

Proof Similar to θ, we define $q'_{\varphi^t}(x) = \frac{p'_{\varphi^t}(x)g_{f,\varphi^t}(x)}{\mathbb{E}_{p'_{\varphi^t}}[g_{f,\varphi^t}(x)]}$. Initially, we have $q'_{\varphi^0}(x) = q_{\theta^0}(x)$ from given conditions. So we assume for the current time t, $q'_{\varphi^t}(x) = q_{\theta^t}(x)$ holds as well. Then from equation (10), we have

$$\tilde{\nabla}_\varphi|_{\varphi=\varphi^{t+1}}(-\sum_{\tau=0}^{T}\lambda^\tau D_{KL}(q'_{\varphi^{t-\tau}}||p'_\varphi) + \eta(\epsilon^2/2 - D_{KL}(p'_{\varphi^t}||p'_\varphi))) \tag{28}$$

$$= \tilde{\nabla}_\theta|_{\theta=\varphi^{-1}(\varphi^{t+1})}(-\sum_{\tau=0}^{T}\lambda^\tau D_{KL}(q_{\theta^{t-\tau}}||p_\theta) + \eta(\epsilon^2/2 - D_{KL}(p_{\theta^t}||p_\theta)))\cdot(\frac{\partial\theta}{\partial\varphi}|_{\theta=\varphi^{-1}(\varphi^{t+1})})^3 \tag{29}$$

$$= 0 \tag{30}$$

Thus we have $\varphi^{t+1} = \varphi(\theta^{t+1})$. From mathematical infuction, our proof finished. ■

A.3. Derivations of SYNCMA

The derivations consist of two parts. The first part is to substitute optimization objevtive $G^t(\theta) = \sum_{\tau=0}^{t-1} \lambda^\tau D_{KL}(q_{\theta^{t-\tau}}||p_\theta)$ with a self-evolved term $M^t(\theta)$ so that the gradient information is completely preserved as equation (11) while the computational costs reduce to the same as IGO, i.e. the cost of computing $\tilde{\nabla}_\theta|_{\theta=\theta^t} L_{\theta^t}(\theta)$. The second part is to derive analytical updates for $\theta^{t+1} = (m^{t+1}, C^{t+1})$ from solving equation (10).

A.3.1. SUBSITITUTION OF $G^t(\theta)$

We start from our target of preserving historical information,

$$\tilde{\nabla}_\theta G^t(\theta) = -\tilde{\nabla}_\theta M^t(\theta) + \tilde{\nabla}_\theta D_{KL}(q_{\theta^t}||p_\theta) \tag{31}$$

We assume $M^t(\theta)$ has the form that present in the main paper, with scalars $\lambda_0, Q_1^t \in \mathbb{R}$ and vectors $s_m^t, s_c^t, Q_2^t, Q_3^t \in \mathbb{R}^n$ all start from zero. We use $\circ$ to denote $v_1 \circ v_2 \equiv v_1 v_2^T + v_2 v_1^T$ for two vectors $v_1, v_2 \in \mathbb{R}^n$.

$$\tilde{\nabla}_m M^t(\theta) = \lambda_0(s_m^t + m^t - m) \tag{32}$$

$$\begin{aligned}\tilde{\nabla}_c M^t(\theta) = &\lambda_0((s_c^t + m^t - m)(s_c^t + m^t - m)^T - C) \\ &+ Q_1^t + Q_2^t \circ m + Q_3^t m m^T\end{aligned} \tag{33}$$

Notice that $G^t(\theta) = \lambda G^{t-1} + D_{KL}(q_{\theta^t}||p_\theta)$, therefore we have,

$$\tilde{\nabla}_\theta M^t(\theta) = \lambda(\tilde{\nabla}_\theta M^{t-1}(\theta) - \tilde{\nabla}_\theta D_{KL}(q_{\theta^{t-1}}||p_\theta)) \tag{34}$$

Mean Solution . Apply assumption (14) for $\tilde{\nabla}_m M^t(\theta)$ yields,

$$\lambda_0(s_m^t + m^t - m) = \lambda\lambda_0(s_m^{t-1} + m^{t-1} - m) + \lambda \sum_i \hat{w}_i(x_i^{t-1} - m) \tag{35}$$

Which straightly gives the only solutions on λ and s_m^t,

$$\lambda = \frac{\lambda_0}{\lambda_0 + 1} \tag{36}$$

$$s_m^t + m^t = \lambda s_m^{t-1} + (1-\lambda) d_w^{t-1} + m^{t-1} \tag{37}$$

Covariance Maxtrix Solution Apply assumption (15) for $\tilde{\nabla}_c M^t(\theta)$ while denoting $Q^t(m) = Q_1^t + Q_2^t \circ m + Q_3^t m m^T$ yields,

$$\begin{aligned}&\lambda_0((s_c^t + m^t - m)(s_c^t + m^t - m)^T - C) + Q^t(m) \\ =\ &\lambda\lambda_0((s_c^{t-1} + m^{t-1} - m)(s_c^{t-1} + m^{t-1} - m)^T - C) + \lambda Q^{t-1}(m) \\ &+ \lambda(\sum_i \hat{w}_i(x_i^{t-1} - m)(x_i^{t-1} - m)^T - C)\end{aligned} \tag{38}$$

To cancel out C on both side, same value of $lambda = \lambda_0/\lambda_0+1$ is derived. Further, we assume the form of update on s_c^t is similar to s_m^t with parameter α and $zeta$ undetermined yet,

$$s_c^t + m^t = \zeta(s_c^{t-1} + m^{t-1}) + \alpha(d_w^{t-1} + m^{t-1}) \tag{39}$$

Thus we arrive at,

$$\begin{aligned}(s_c^t + m^t - m)(s_c^t + m^t - m)^T = \zeta^2(s_c^{t-1} + m^{t-1} - m)(s_c^{t-1} + m^{t-1} - m)^T \\ + \alpha^2(d_w^{t-1} + m^{t-1} - m)(d_w^{t-1} + m^{t-1} - m)^T \\ + (\alpha + \zeta - 2)m \circ (\zeta(s_c^{t-1} + m^{t-1}) + \alpha(d_w^{t-1} + m^{t-1})) \\ + (\alpha - 1)(\zeta - 1)mm^T \\ + \alpha\zeta(d_w^{t-1} + m^{t-1}) \circ (s_c^{t-1} + m^{t-1})\end{aligned} \tag{40}$$

Notice that for the second term of Eq.40, we have,

$$\begin{aligned}&(d_w^{t-1} + m^{t-1} - m)(d_w^{t-1} + m^{t-1} - m)^T \\ &= \mathbb{E}_{q_{\theta^{t-1}}}[x - m]\mathbb{E}_{q_{\theta^{t-1}}}[x - m]^T && (41) \\ &= \mathbb{E}_{q_{\theta^{t-1}}}[(x - \mathbb{E}_{q_{\theta^{t-1}}}[x] + \mathbb{E}_{q_{\theta^{t-1}}}[x] - m)(x - \mathbb{E}_{q_{\theta^{t-1}}}[x] + \mathbb{E}_{q_{\theta^{t-1}}}[x] - m)^T] && (42) \\ &\quad - \mathbb{E}_{q_{\theta^{t-1}}}[(x - \mathbb{E}_{q_{\theta^{t-1}}}[x])(x - \mathbb{E}_{q_{\theta^{t-1}}}[x])^T] \\ &= \mathbb{E}_{q_{\theta^{t-1}}}[(x - m)(x - m)^T] - \mathbb{E}_{q_{\theta^{t-1}}}[(x - \mathbb{E}_{q_{\theta^{t-1}}}[x])(x - \mathbb{E}_{q_{\theta^{t-1}}}[x])^T] && (43) \\ &= \sum_i \hat{w}_i(x_i^{t-1} - m)(x_i^{t-1} - m)^T - \sum_i \hat{w}_i(d_i^{t-1} - d_w^{t-1})(d_i^{t-1} - d_w^{t-1})^T && (44)\end{aligned}$$

Thus, to meet Eq.38, we arrive at updates for $s_c^t, Q_1^t, Q_2^t, Q_3^t$ with $\zeta = \sqrt{\lambda}, \alpha = \sqrt{1 - \lambda}$,

$$\begin{aligned}s_c^t + m^t &= \zeta s_c^{t-1} + \alpha d_w^{t-1} + (\zeta + \alpha)m_{t-1} && (45) \\ Q_1^t &= \lambda Q_1^{t-1} + \lambda \sum \hat{w}_i(d_i^{t-1} - d_w^{t-1})(d_i^{t-1} - d_w^{t-1})^T && (46) \\ &\quad - \lambda_0\alpha\zeta(d_w^{t-1} + m^{t-1}) \circ (s_c^{t-1} + m^{t-1}) \\ Q_2^t &= \lambda Q_2^{t-1} - \lambda_0(\zeta + \alpha - 2)(\zeta * (s_c^{t-1} + m^{t-1}) + \alpha * (d_w^{t-1} + m^{t-1})) && (47) \\ Q_3^t &= \lambda Q_3^{t-1} - \lambda_0(\zeta - 1)(\alpha - 1) && (48)\end{aligned}$$

A.3.2. DERIVATION OF θ^{t+1}

Substitute the $G^t(\theta)$ in the natural Lagrange condition (10) gives equation,

$$\begin{aligned}0 &= \tilde{\nabla}_\theta|_{\theta=\theta^{t+1}}(-G^t(\theta) + \eta(\epsilon^2/2 - D_{KL}(p_{\theta^t}||p_\theta))) && (49) \\ &= \tilde{\nabla}_\theta|_{\theta=\theta^{t+1}}(M^t(\theta) - D_{KL}(q_{\theta^t}||p_\theta) - [\eta_m, \eta_c]^T D_{KL}(p_{\theta^t}||p_\theta)) && (50)\end{aligned}$$

Apply proposition 5 on above equation, we now arrive at equations for θ^{t+1},

$$\sum_i(\hat{w}_i^t + \frac{\eta_m}{N})(x_i^t - m^{t+1}) + \tilde{\nabla}_m M^t(\theta) = 0 \tag{51}$$

$$\sum_i(\hat{w}_i^t + \frac{\eta_c}{N})((x_i^t - m^{t+1})(x_i^t - m^{t+1})^T - C^{t+1}) + \tilde{\nabla}_c M^t(\theta) = 0 \tag{52}$$

Solving equations above straightly give updates with $z_m = \eta_m + \lambda_0 + 1$, $z_c = \eta_c + \lambda_0 + 1$, $\beta^t = \frac{1}{z_m}(d_w^t + \lambda_0 s_m^t)$ for brevity sake.

$$m^{t+1} = m^t + \beta^t \tag{53}$$

$$C^{t+1} = \frac{\eta_c}{z_c}(C^t + \beta^t(\beta^t)^T) + \frac{\lambda_0}{z_c}(s_c^t - \beta^t)(s_c^t - \beta^t)^T + \frac{1}{z_c}(\sum_i \hat{w}_i(d_i^t - \beta^t)(d_i^t - \beta^t)^T + Q_1^t + Q_2^t \circ m^t + Q_3^t m^t (m^t)^T) \tag{54}$$

A.4. Proof and discussions on Proposition 6

Proof We apply the first condition to equations (51, 52), which gives the updates for θ^{t+1} with $D_w^t = \sum_i \hat{w}_i^t (x_i^t - m^t)(x_i^t - m^t)^T$,

$$m^{t+1} = m^t + \frac{1}{z_m} d_w^t \tag{55}$$

$$C^{t+1} = \frac{\eta_c}{z_c} C^t + \frac{1}{z_c} D_w^t + \frac{\lambda_0}{z_c} s_c^t (s_c^t)^T - \frac{\lambda_0}{z_c z_m}(d_w^t \circ (s_c^t)) - \frac{1}{z_c z_m}(2 - \frac{z_c}{z_m}) d_w^t (d_w^t)^T \tag{56}$$

Then under the second condition, the last two terms in the update of C^{t+1} should be discarded. The rest part coincide with CMA-ES up to an external learning rate difference. ■

Appendix B. Experiments

We implement SYNCMA using PyPop7 (Duan et al., 2022)[2]. All baselines except TuRBO and TR-CMA-ES are also implemented with this library. Except for the hyperparameters mentioned in the main paper, all evolutionary based algorithms use default options. In SYNCMA, we use constant learning rate $\sigma = 0.1$ in order to match the initial learning rate in other CMA optimizers to sample with $\mathcal{N}(m, \sigma\Sigma)$. As CMA optimizers also use σ as the update fraction for mean, we set η_m such that $1/z_m = 0.1$ accordingly. η_c is set two times of the corresponding value in CMA-ES, as several tunning techniques used in fine-tuned version of CMA-ES may let its actual corresponding value larger. For TuRBO, we refer to the implementation of BoTorch and replicate the original algorithm. We set the trust region number as one and keep all hyperparameter same as the original implementation.[3]

All experiments are held on Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz, except for TuRBO method, a NVIDIA A100 is used. No tuning is applied for all optimizers including SYNCMA.

B.1. Mujoco Locomotion Task

For Mujoco locomotion benchmarks, we refer to ARS[4] to model the task as a sampling problem, rollout is set to 1 for simplicity. Full optimization procedure for all 5 tasks are provided in 6.

2. https://github.com/Evolutionary-Intelligence/pypop
3. https://botorch.org/tutorials/turbo_1
4. https://github.com/modestyachts/ARS

B.2. Rover Planning Task

For rover planning task, we refer to nevergrad[5] to test. Optimization procedure is provided in 6(*f*).

B.3. Synthetic Functions

For synthetic functions, we directly use the benchmarks provided by PyPop7, with formulation shown in Table 2. Full contents of near optimal performance and the average efficiency rank, i.e. the first time hitting time into 0.5 value, for CMA optimizers under the same budget are provided in tables 3 and 5, other optimizers are excluded as their poor performance and linewidth limit. The optimization procedure of all 10 functions for SYNCMA with different λ_0 and other baselines are provided in figures 3, 4, 5.

It can be seen that as dimension increasing, the average performance of SYNCMA is getting worse. This align with the observation on dimensionality in ablation study that higher dimension generally yields higher λ_0.

Table 2: 10 different synthetic functions. Selected from classic test function for global optimization, including different rugged characteristics like multi-modality, high condition number and different optima landscape. The optima all scale to $x^* = 0, f(x^*) = 0$. For the brevity sake, $w_i = 1 + \frac{x_i+1}{4}$ in LevyMontalvo function, $z_i = x_i^2 + x_{i+1}^2$ in Schaffer function.

Name	Expression
Sphere	$\sum_i x_i^2$
Discus	$10^6 x_0 + \sum_{i\geq 1} x_i$
Schwefel	$\sum_i x_i + \prod_i x_i$
DiffPowers	$\sum_i x^{2+4i/n}$
Bohachevsky	$\sum_{i\leq n-1} 0.7 + x_i^2 + 2 * x_{i+1}^2 - 0.3\cos(3\pi x_i) - 0.4\cos(4\pi x_{i+1})$
LevyMontalvo	$\frac{\pi}{n}(10\sin^2(\pi w_0) + (w_{n-1}-1)^2 + \sum_{i<n-1}(w_i-1)^2(1+10\sin^2(\pi w_{i+1})))$
Rastrigin	$10n + \sum_i x_i^2 - 10\cos(2\pi x_i)$
Ackley	$-20\exp(-0.2(\frac{1}{n}\sum_i x_i^2)^{0.5}) - \exp(\frac{1}{n}\sum_i \cos(2\pi x_i)) + 20 + e$
Schaffer	$\sum_{i<n-1} z_i^{0.25} * (\sin^2(50 * z_i^{0.1}) + 1)$
Rosenbrock	$100\sum_{i\geq 1}(x_i - x_{i-1}^2)^2 + \sum_{i<n-1}(x_i - 1)^2$

B.4. Ablation Study

All the data provided in appendix actually contains different version of SYNCMA with $\lambda_0 = \{0, 1, 2, 4\}$, so they naturally consist the ablation study. The three observations can be verified thereby. Through every experiment on each λ_0, we monitor the behavior of SYNCMA and do not find any degreneration happen.

5. https://github.com/facebookresearch/nevergrad

Table 3: Near optimal performance of CMA optimizers on 32d synthetic functions(lower is better). We show the median best performance of 10,000 evaluations over 100 trials. Numbers in brackets indicates the evaluation number need for algorithms to achieve better than 0.5.

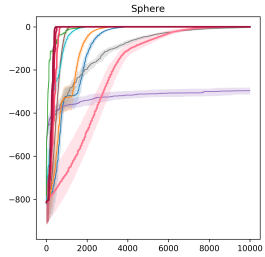
Function	DD-CMA	CMA-ES	TR-CMA-ES	SynCMA_1	SynCMA_2	SynCMA_4
Sphere	0.0(3665)	0.0(4612)	0.1(2405)	0.0(855)	0.0(555)	0.0(455)
Discus	0.0(5223)	52.8	419.6	0.1(6363)	0.0(4437)	0.0(2568)
Schwefel	0.0(5325)	0.0(5124)	0.1(3074)	0.0(704)	0.0(519)	0.0(449)
DiffPowers	0.0(3703)	0.0(5585)	0.1(3720)	0.0(704)	0.0(519)	0.0(450)
Bohachevsky	0.0(5420)	0.0(7115)	0.4(4428)	0.2(8163)	0.1(6340)	0.0(5106)
LevyMontalvo	0.0(5153)	0.0(4030)	0.1(2506)	0.0(1350)	0.0(1092)	0.1(1094)
Rastrigin	58.1	194.8	15.9	1.0	0.2(8758)	0.1(7372)
Ackley	0.0(4849)	0.0(5850)	0.1(2946)	0.1(2733)	0.0(2133)	0.0(1134)
Schaffer	61.3	71.3	2.2	6.6	4.8	3.8
Rosenbrock	28.0	29.9	0.1(7251)	29.3	29.2	29.6
Ave_Rank	5	6	4	3	2	1

Table 4: Near optimal performance of CMA optimizers on 64d synthetic functions(lower is better). We show the median best performance of 50,000 evaluations over 20 trials. Numbers in brackets indicates the evaluation number need for algorithms to achieve better than 0.5.

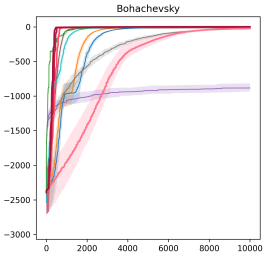
Function	DD-CMA	CMA-ES	TR-CMA-ES	SynCMA_1	SynCMA_2	SynCMA_4
Sphere	0.0(10047)	0.0(13825)	0.0(7185)	0.0(4691)	0.0(3938)	0.0(1047)
Discus	0.0(12748)	0.0(43265)	85.9	0.0(27662)	0.0(18820)	0.0(10484)
Schwefel	0.0(16820)	0.0(16134)	0.0(9905)	0.0(1548)	0.0(1157)	0.0(929)
DiffPowers	0.0(10164)	0.0(18503)	0.0(12040)	0.0(1548)	0.0(1158)	0.0(937)
Bohachevsky	0.0(14605)	0.0(20369)	0.0(11865)	0.3(44088)	0.0(32247)	0.0(23314)
LevyMontalvo	0.0(9087)	0.0(9901)	0.0(5515)	0.0(2522)	0.0(2318)	0.0(2486)
Rastrigin	17.9	21.4	22.4	1.3	0.2(42696)	0.0(31565)
Ackley	0.0(11757)	0.0(15620)	0.0(7869)	0.0(9800)	0.0(7567)	0.0(4549)
Schaffer	23.3	0.4(49400)	0.3(32701)	12.3	8.2	4.9
Rosenbrock	53.8	57.8	0.0(21543)	59.8	60.1	60.9
Ave_Rank	5	6	3	4	2	1

Table 5: Near optimal performance of CMA optimizers on 128d synthetic functions(lower is better). We show the median best performance of 100,000 evaluations over 20 trials. Numbers in brackets indicates the evaluation number need for algorithms to achieve better than 0.5.

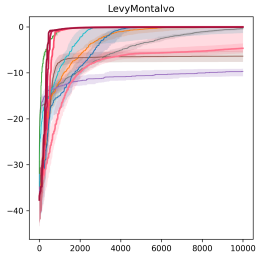
Function	DD-CMA	CMA-ES	TR-CMA-ES	SynCMA_1	SynCMA_2	SynCMA_4
Sphere	0.0(39745)	0.0(27751)	0.0(21428)	0.2(54617)	0.1(40403)	0.0(25428)
Discus	395.8	0.0(32544)	1496.4	1.0	0.4(93272)	0.1(58425)
Schwefel	0.0(51681)	0.0(59947)	0.0(31407)	0.1(3610)	0.1(2600)	0.1(2085)
DiffPowers	0.0(62675)	0.0(28185)	0.0(41019)	0.0(3626)	0.0(2621)	0.0(2123)
Bohachevsky	0.0(57138)	0.0(39401)	0.0(31801)	9.8	4.8	1.2
LevyMontalvo	0.0(26924)	0.0(20332)	0.0(14875)	0.0(5672)	0.0(5404)	0.0(5344)
Rastrigin	107.6	23.0	33.3	42.6	19.9	5.0
Ackley	0.0(41488)	0.0(29714)	0.0(21994)	0.3(46327)	0.2(33839)	0.1(20330)
Schaffer	4.3	0.7	0.2(91494)	45.1	37.7	27.9
Rosenbrock	124.3	120.5	0.0(68532)	146.6	134.8	127.8
Ave_Rank	5	3	2	6	4	1



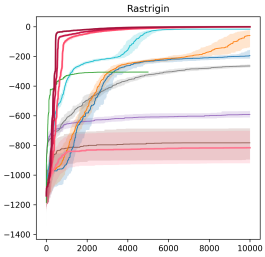
(*a*) Sphere



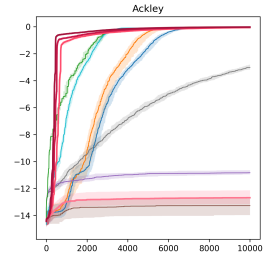
(*b*) Bohachevsky



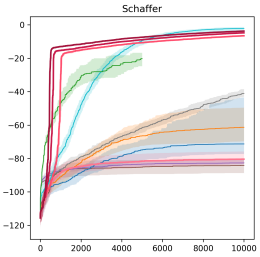
(*c*) LevyMontalvo



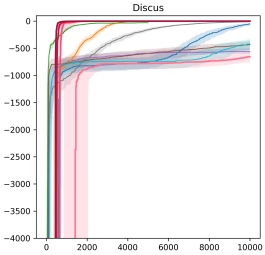
(*d*) Rastrigin



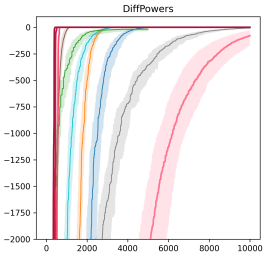
(*e*) Ackley



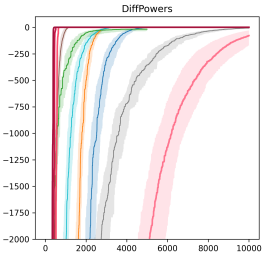
(*f*) Schaffer



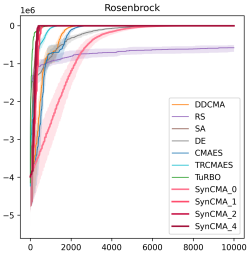
(*g*) Discus



(*h*) Schwefel

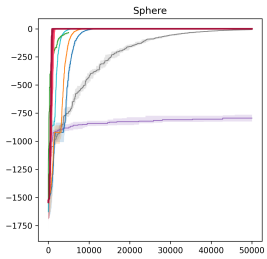


(*i*) DiffPowers

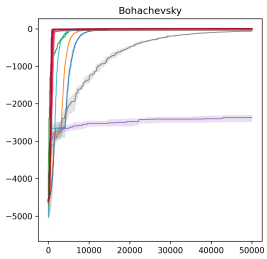


(*j*) Rosenbrock

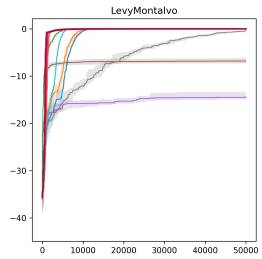
Figure 3: Optimization procedure in 10 tests function with dimension $n = 32$ over 100 trails with 10000 evaluations.



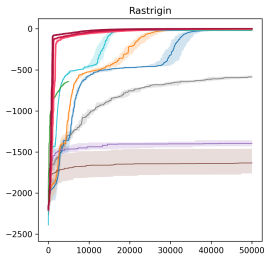
(*a*) Sphere



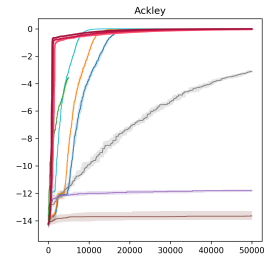
(*b*) Bohachevsky



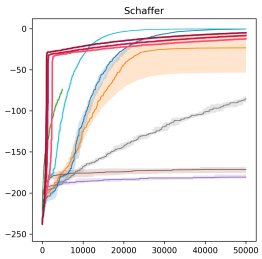
(*c*) LevyMontalvo



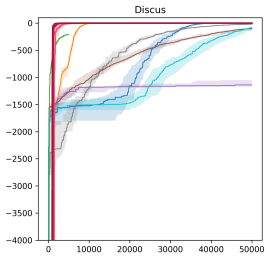
(*d*) Rastrigin



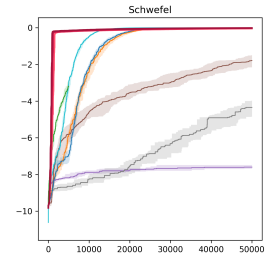
(*e*) Ackley



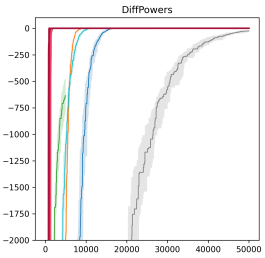
(*f*) Schaffer



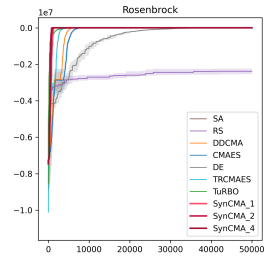
(*g*) Discus



(*h*) Schwefel

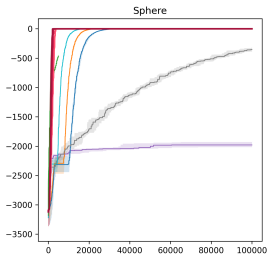


(*i*) DiffPowers

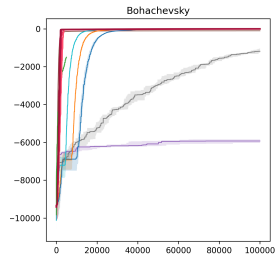


(*j*) Rosenbrock

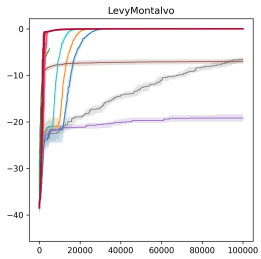
Figure 4: Optimization procedure in 10 tests function with dimension $n = 64$ over 100 trails with 50000 evaluations.



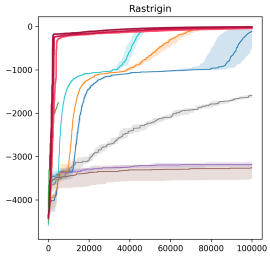
(*a*) Sphere



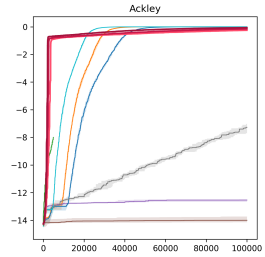
(*b*) Bohachevsky



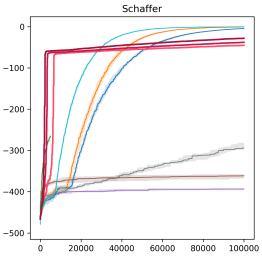
(*c*) LevyMontalvo



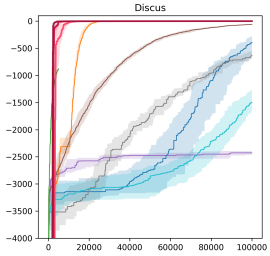
(*d*) Rastrigin



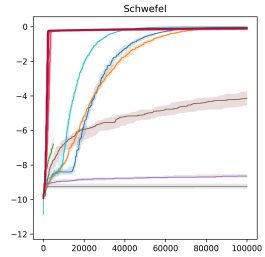
(*e*) Ackley



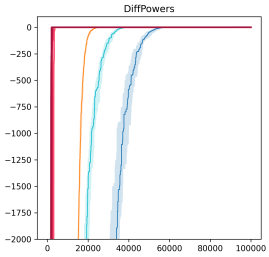
(*f*) Schaffer



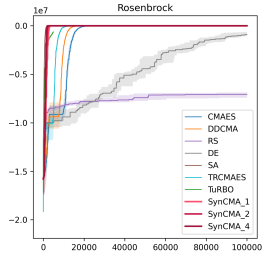
(*g*) Discus



(*h*) Schwefel

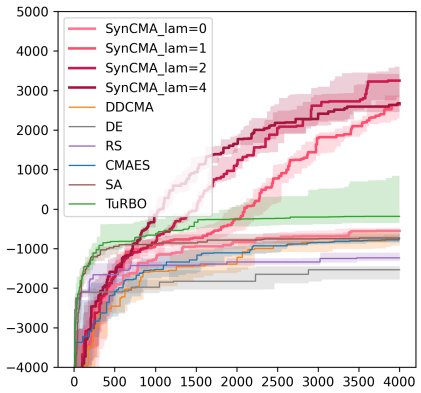


(*i*) DiffPowers

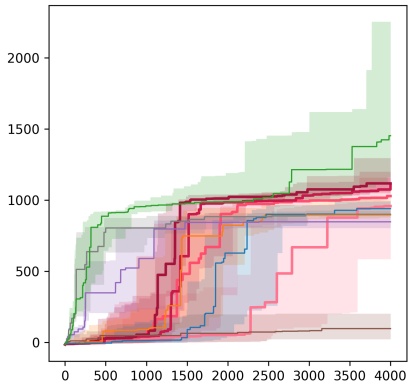


(*j*) Rosenbrock

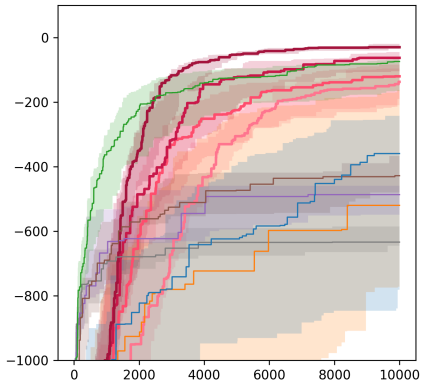
Figure 5: Optimization procedure in 10 tests function with dimension $n = 128$ over 100 trails with 100000 evaluations.



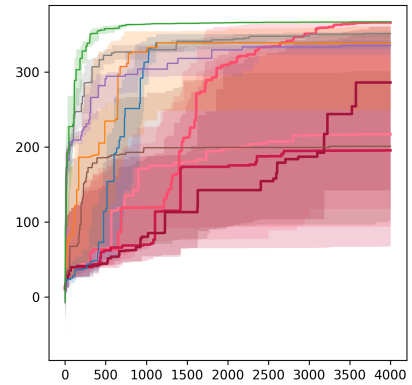
(*a*) Half-Cheetah(102d)



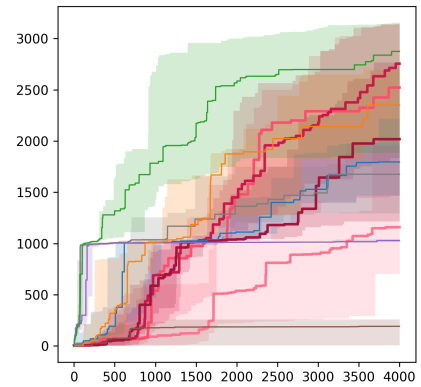
(*b*) Walker(102d)



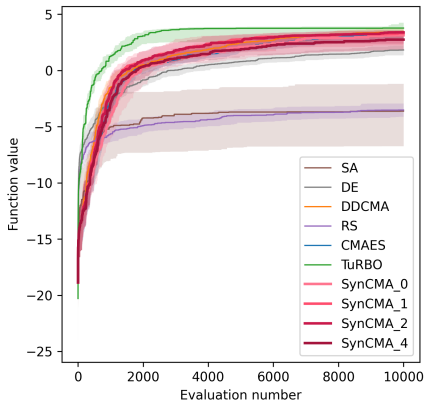
(*c*) Ant(888d)



(*d*) Swimmer(16d)



(*e*) Hopper(33d)



(*f*) Rover planning(60d)

Figure 6: Optimization performance on 5 Mujoco locomotion tasks and the rover planing task