Journal of Machine Learning Research 25 (2024) 1-44 Submitted 5/22; Revised 12/23; Published 5/24

Robust Black-Box Optimization for Stochastic Search and Episodic Reinforcement Learning

Maximilian Hüttenrauch MAXIMILIAN.HUETTENRAUCH@KIT.EDU
Gerhard Neumann GERHARD.NEUMANN@KIT.EDU
Department of Computer Science
Karlsruhe Institute of Technology
Karlsruhe

Editor: Alekh Agarwal

Abstract

Black-box optimization is a versatile approach to solve complex problems where the objective function is not explicitly known and no higher order information is available. Due to its general nature, it finds widespread applications in function optimization as well as machine learning, especially episodic reinforcement learning tasks. While traditional black-box optimizers like CMA-ES may falter in noisy scenarios due to their reliance on ranking-based transformations, a promising alternative emerges in the form of the Model-based Relative Entropy Stochastic Search (MORE) algorithm. MORE can be derived from natural policy gradients and compatible function approximation and directly optimizes the expected fitness without resorting to rankings. However, in its original formulation, MORE often cannot achieve state of the art performance. In this paper, we improve MORE by decoupling the update of the search distribution's mean and covariance and an improved entropy scheduling technique based on an evolution path resulting in faster convergence, and a simplified model learning approach in comparison to the original paper. We show that our algorithm performs comparable to state-of-the-art black-box optimizers on standard benchmark functions. Further, it clearly outperforms ranking-based methods and other policy-gradient based black-box algorithms as well as state of the art deep reinforcement learning algorithms when used for episodic reinforcement learning tasks.

Keywords: black-box optimization, stochastic search, derivative-free optimization, evolution strategies, episodic reinforcement learning

1. Introduction

Stochastic-Search algorithms (Spall, 2005) are problem independent algorithms well-suited for black-box optimization (BBO) (Larson et al., 2019) of an objective function. They only require function evaluations and are used when the objective function cannot be modeled analytically and no gradient information is available. This is often the case for real world problems such as robotics (Chatzilygeroudis et al., 2017) where the objective function describes the outcome of a task, medical applications (Winter et al., 2008), or forensic identification (Ibáñez et al., 2009).

Typically, these algorithms maintain a search distribution over the optimization variables of the objective function. Solution candidates are sampled, evaluated, and the parameters of the search distribution (e.g. mean and covariance for Gaussian search distributions) are then updated towards a more promising direction. This process is repeated until a satisfactory solution quality is found or a pre-defined budget of objective function evaluations is reached.

In this paper, we re-introduce Model-based Relative Entropy Stochastic Search (MORE) (Abdolmaleki et al., 2015), a versatile, general purpose stochastic-search optimization algorithm. Using insights from reinforcement learning (RL) and information-theoretic trust-regions, it aims to update the parameters of a Gaussian search distribution in the direction of the natural gradient (Kakade, 2001). To this end, MORE uses compatible function approximation (Sutton et al., 1999; Pajarinen et al., 2019) and learn a quadratic surrogate model of the objective function and bound the Kullback-Leibler (KL) divergence between subsequent search distributions. In its original formulation, a bound on the Kullback-Leibler divergence and a bound on the loss of entropy between the old and new search distribution additionally acts as an exploration-exploitation trade-off to prevent pre-mature convergence of the algorithm.

Most other successful stochastic search algorithms make use of rankings of objective function evaluations for updating the search distribution's parameters (Hansen, 2016; Wierstra et al., 2014; Rubinstein and Kroese, 2004). While the use of rankings is well studied for deterministic objective function evaluations, its effects in the case of stochastic objective function evaluations, which are, for example, common in episodic reinforcement learning, are less well understood. We analyze these ranking based algorithms for this stochastic evaluation case and uncover their limitations to solve such tasks. In contrast, MORE directly optimizes the expected objective function values without ranking transformations of the objective function evaluations and can therefore also be applied for such domains.

However, the original MORE approach suffers from (a) the need for conservative KL bounds to keep the covariance updates stable, (b), a fixed entropy decreasing schedule resulting in suboptimal exploration and slow convergence, and (c), it employs an unnecessary complicated and inaccurate model learning method. We aim to fix these issues by (a) splitting the KL divergence into separate updates for the mean and covariance with separate trust region bounds, (b) introducing an adaptive entropy schedule based on an evolution path, and (c) using ordinary least squares with appropriate data pre-processing techniques to simplify the model learning process. We call our new algorithm **C**oordinate-**A**scent MORE with **S**tep Size Adaptation or CAS-MORE for short.

We empirically evaluate our algorithm on simulated robotics tasks, as well as a set of benchmark optimization functions. While we are competitive with state-of-the-art algorithms such as CMA-ES on the benchmark optimization functions, the RL experiments clearly show the strength of MORE.

2. Related Work

In the following sections, we describe in more depth current state of the art algorithms from the field of black-box optimization and review common design choices and algorithmic procedures. In addition, we highlight current step- and episode-based reinforcement learning approaches along with their benefits and drawbacks and discuss trust-region based reinforcement learning algorithms.

2.1 Evolutionary Strategies and Black-Box Optimization

The MORE algorithm can be seen as an instance of Evolution Strategies (Beyer and Schwefel, 2002) from the broader class of Evolutionary Algorithms. The usual procedure involves sampling from a search distribution $\pi(\boldsymbol{x};\theta)$ with parameters θ, evaluating the candidates $\boldsymbol{x}_k$ on an objective function $f(\boldsymbol{x})$, $f:\mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{x} \mapsto f(\boldsymbol{x})$, and the goal is to either find an individual $\boldsymbol{x}^*$ that maximizes f or a distribution π^* that maximizes the optimization objective $J = \mathbb{E}_{\boldsymbol{x}\sim\pi}[f(\boldsymbol{x})]$.

2.1.1 Ranking-Based Algorithms

Instead of directly incorporating function values into the optimization process, many algorithms apply a ranking based transformation , i.e., a monotonous mapping of function values sorted by fitness to some pre-defined fixed values. Specifically, for a population $\{\boldsymbol{x}_{i:\lambda} \mid i = 1,\ldots,\lambda\} = \{\boldsymbol{x}_i \mid i = 1,\ldots,\lambda\}$ sorted by function values such that $f(\boldsymbol{x}_{1:\lambda}) \le f(\boldsymbol{x}_{2:\lambda}) \le \cdots \le f(\boldsymbol{x}_{\lambda:\lambda})$, the function values are substituted by weight values $w_1 <,\ldots,< w_\lambda$, before updating parameters. While this makes the optimization process invariant to certain transformations and adds to the robustness of algorithms, it also changes the optimization objective into $\theta^* = \arg\max_\theta \sum_i w_i \log \pi(\boldsymbol{x}_i;\theta)$. As we will show in our experiments, this change can have severe downsides in reinforcement learning problems. In non-deterministic problems, it leads to an over- or underestimation of a sample's performance and requires averaging over several sample evaluations in order to obtain a reliable estimate for the ranking (Hansen et al., 2008; Heidrich-Meisner and Igel, 2009).

The cross-entropy method (Rubinstein and Kroese, 2004; Botev et al., 2013; Amos and Yarats, 2020) is one of the simplest representatives of ranking based algorithms. It evolves the search distribution by only incorporating an elite set of samples into the next generation which can be seen as a rank-based update.

The Covariance Matrix Adaptation - Evolution Strategy (CMA-ES) (Hansen, 2016) performs well established heuristics to update the mean and covariance matrix of a Gaussian search distribution as an interpolation of the weighted sample mean and covariance and the old mean and covariance. The weights of the samples are chosen according to the ranking of the samples. CMA-ES also introduces the evolution path, an exponentially smoothed cumulative sum of previous update steps, which acts as a momentum term, similar to gradient based optimizers (Li and Zhang, 2016; Ruder, 2016). Building on the basic CMA-ES approach, many derivative algorithms exist, mainly aiming at improving sample efficiency. Notable extensions are those which include restarts with increasing population sizes (Auger and Hansen, 2005), restarts that alternate between large and small population sizes (Hansen, 2009), a version with decreasing step-sizes (Loshchilov et al., 2012a), or incorporating second order information into the update of the covariance matrix (Auger et al., 2004).

Another class of algorithms using rank-based fitness shaping are those from the family of Natural Evolution Strategies (NES) (Wierstra et al., 2014). Instead of updating the search distribution parameters with heuristics, NES follow a sample-based search gradient based on the same objective as MORE. Yet, due to the use of rankings, the connection to the natural gradient of the original expected reward objective is lost. We describe the relation to MORE in more detail in the next section.

Contrary to these algorithms, MORE does not discard poorly performing candidates in its optimization process and only applies common data pre-processing techniques such as standardization before feeding them to the learning algorithm. There also exist variations of the CMA-ES using surrogate models such as (Loshchilov et al., 2012b) or (Hansen, 2019). Whereas these algorithms use the surrogate to generate approximate function values, which are then again used to generate a ranking, MORE directly uses the model parameters to update the search distribution parameters.

2.1.2 Natural Gradients for Black-Box Optimization

The natural gradient is often used in optimization as it has shown to be more effective when a parameter space has a certain underlying structure (Amari, 1998). Important algorithms belong to the family of Natural Evolution Strategies (NES) (Wierstra et al., 2014). They use a Taylor approximation of the KL-divergence between subsequent updates to estimate a search gradient in the direction of the natural gradient. Instead of information theoretic trust-regions on the update as used in MORE, they use either fixed (Sun et al., 2009; Glasmachers et al., 2010) or heuristically updated learning rates (Wierstra et al., 2014). NES also uses ranking-based fitness shaping. The ranking is required to improve the robustness of the algorithm, yet, it also changes the objective and the search direction does not correspond to the natural gradient anymore. In contrast, MORE is inherently robust due to the used trust-regions and therefore, does not require a ranking-based transformation of the rewards. The ROCK$*$ algorithm presented in (Hwangbo et al., 2014) uses the natural gradient on a global approximation of the objective generated with kernel regression.

2.1.3 Other Black-Box Optimization Approaches

There exists a wide variety of stochastic search algorithm classes for black-box optimization, each with their own benefits and drawbacks. Classic algorithms such as Nelder-Mead (Nelder and Mead, 1965) use a simplex to find the minimum of a function. Bayesian optimization techniques such as Gaussian processes, for example, aim at finding global optima in low dimensional problem domains (Osborne et al., 2009). However, they suffer from high computation time and scale poorly with problem dimensionality and data points. Other directions are genetic algorithms (Holland, 1992) which are easy to implement but suffer from the need for good heuristics and random search (Zabinsky, 2010; Price, 1983). While both approaches can yield good results, they are not computationally efficient.

2.2 Reinforcement Learning

As MORE is based on the policy search objective, we also review some of the recent work from the domain of reinforcement learning. Here, the objective is based on the reward function defining the task to be solved. A detailed introduction to the problem domain can be found in Section 3.

2.2.1 Step-based Trust-Region Reinforcement Learning Algorithms

Updates bounded by a trust-region are a common approach in the reinforcement learning literature. One of the first deep reinforcement learning algorithms to successfully make use of a trust-region update is TRPO (Schulman et al., 2015) which enforces a bound on an average KL divergence trust-region. Similar sample-based approximations of the trust-region are also employed by recent deep reinforcement learning techniques such as Maximum A-Posteriori Policy Optimization (MPO) (Abdolmaleki et al., 2018b). Compatible function approximation for deep reinforcement learning has been explored in (Pajarinen et al., 2019). An improved way of enforcing trust-regions directly in the policy function by using differentiable trust-region layers (TRPLs) has been introduced in (Otto et al., 2021).

2.2.2 Episode-based Reinforcement Learning

Episode-based reinforcement learning algorithms make use of black-box optimization techniques to optimize the expected return of an entire trajectory. These algorithms excel when tasks are defined through sparse and non-Markovian rewards, whereas step-based approaches typically fail in these cases. Due to the use of sparse rewards, learned behaviors are more energy efficient and less sensitive to high action costs compared to policies learned by step-based reinforcement learning algorithms. On the downside, the improved performance often comes at a higher sample complexity (Otto et al., 2023).

A closely related method to MORE is the Relative Entropy Policy Search (REPS) for step-based reinforcement learning (Peters et al., 2010) and its episodic formulation (albeit derived in a contextual setting) in (Kupcsik et al., 2013). Both use samples to approximate the objective and the trust-regions which does not guarantee that the KL trust-region is enforced for the new parametric policy in practice. Layered direct policy search (LaDIPS) (End et al., 2017) extends MORE to the contextual setting using a linear mapping from the context to the mean. Additionally, the usage of natural gradients can be derived by using a second order approximation of the KL-divergence, resulting in the Fisher information matrix used in NES.

Recently, Otto et al. (2023) extended TRPLs to the episodic case and use a policy gradient method for learning movement primitives that allow a non-linear mapping from a context vector to the motion primitive parameters. They show that other algorithms such as PPO (Schulman et al., 2017) are unsuitable as they suffer from the higher dimensional action space induced by the motion primitives. In contrast, we focus on the non-contextual case where the policy is a Gaussian distribution with full covariance matrix. We show that in this case, the natural gradient updates from MORE clearly outperform the trust-region policy gradient updates from TRPL, indicating that extending the exact natural gradient updates from MORE to deep neural networks is an interesting direction for future work.

2.3 Broader Scope

The MORE algorithm finds application in a variety of fields, such as variational inference (Arenz et al., 2018) or density estimation (Becker et al., 2019). In the context of trajectory optimization, ideas from the MORE algorithm are explored in the MOTO algorithm (Akrour et al., 2018).

3. Preliminaries

Before we introduce our new algorithm, we present in detail the problem setting and the original MORE algorithm. Additionally, we provide another view on the algorithm derived from compatible feature approximation and its relation to natural gradient optimization.

3.1 Problem Setting

The goal is to find a distribution $\pi(\boldsymbol{x};\theta)$, parameterized by θ, over variables $\boldsymbol{x} \in \mathbb{R}^n$ that maximizes [1] the expected value $\mathcal{F}(\boldsymbol{x}) = \mathbb{E}[F(\boldsymbol{x},\, \xi(\omega))]$ of a noisy objective function F where $\xi(\omega)$ is a typically unknown noise process. In the episodic reinforcement learning case (Deisenroth et al., 2013; Otto et al., 2023), F describes the episodic return. There can be multiple sources of noise $\xi(\omega)$, such as noise in the dynamics or the state observations of the given system. We frame the stochastic search objective as maximizing the expected fitness function, i.e.,

$$\max_{\pi} J = \max_{\pi} \mathbb{E}_{\boldsymbol{x}\sim\pi}[\mathcal{F}(\boldsymbol{x})] \tag{1}$$

which is a well known objective in policy search (Deisenroth et al., 2013). We consider the black-box scenario where we only have access to the noisy function evaluations $f(\boldsymbol{x}) = F(\boldsymbol{x},\, \zeta)$, $\zeta \sim \xi(\omega)$, of the objective function, i.e., the expected function values $\mathcal{F}(\boldsymbol{x})$ are unknown and no gradients of the objective function are available.

3.2 Model-Based Relative Entropy Stochastic Search

MORE (Abdolmaleki et al., 2015) is an iterative stochastic search algorithm where the search distribution $\pi_t(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ in each iteration t is modelled as a multivariate Gaussian distribution. The parameters to be optimized are $\theta = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ and the optimization problem is given by

$$\begin{aligned} \underset{\pi}{\text{maximize}} \quad & \int_{\boldsymbol{x}} \pi(\boldsymbol{x})\mathcal{F}(\boldsymbol{x})\mathrm{dx} \\ \text{subject to} \quad & \mathrm{KL}(\pi \parallel \pi_t) \leq \epsilon, \\ & H(\pi) \geq \beta, \\ & \int_{\boldsymbol{x}} \pi(\boldsymbol{x})\mathrm{dx} = 1 \end{aligned} \tag{2}$$

where ϵ and β are hyper-parameters controlling the exploration-exploitation trade-off, $H(\pi) = \int_{\boldsymbol{x}} \pi(\boldsymbol{x}) \log \pi(\boldsymbol{x})\mathrm{dx}$ denotes the Shannon entropy of the search distribution, and $\mathrm{KL}(\pi \parallel \pi_t) = \int_{\boldsymbol{x}} \pi(\boldsymbol{x}) \log \frac{\pi(\boldsymbol{x})}{\pi_t(\boldsymbol{x})}\mathrm{dx}$ is the KL divergence between the current and new search distribution.

MORE optimizes the objective from Equation (2) by substituting the expected objective value with a learned quadratic approximation

$$\mathcal{F}(\boldsymbol{x}) \approx \hat{\mathcal{F}}(\boldsymbol{x}) = -1/2\, \boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}}\boldsymbol{a} + a_0. \tag{3}$$

Using a quadratic surrogate and a Gaussian search distribution, the optimal solution in each iteration would be to set the mean of the search distribution to the optimum of the surrogate

1. Equivalently, a cost can be minimized by maximizing the negative objective function.

and collapse the search distribution to a point estimate which would prevent it from further exploration. To control the exploration-exploitation trade-off, additional constraints on the updated search distribution need to be introduced. First, the KL-divergence between the current search distribution and the solution of the optimization problem is upper bounded which ensures that the mean and covariance only slowly change and the approximate model is not over-exploited. Furthermore, an additional lower bound on the entropy is introduced to prevent premature convergence.

Closed Form Updates The optimization problem can be solved in closed form using the method of Lagrangian multipliers. The dual function is given by

$$g(\eta, \omega) = \eta\epsilon - \omega\beta + (\eta + \omega) \log \left(\int \pi_t(\boldsymbol{x})^{\frac{\eta}{\eta+\omega}} \exp \left(\frac{\mathcal{F}(\boldsymbol{x})}{\eta + \omega} \right) \mathrm{dx} \right)$$

with Lagrangian multipliers η and ω. The new search distribution is then given in terms of the Lagrangian multipliers as

$$\pi(\boldsymbol{x}) \propto \pi_t(\boldsymbol{x})^{\frac{\eta}{\eta+\omega}} \exp \left(\frac{\mathcal{F}(\boldsymbol{x})}{\eta + \omega} \right).$$

Analytic Solution The use of a quadratic model is beneficial for two reasons. First, it is expressive enough to solve a wide range of problems while being computationally easy to obtain. Second, quadratic features are the compatible features of the Gaussian distribution which make the integrals tractable. As shown in Section 3.3, the use of quadratic features also allows us to obtain exact natural gradient updates. The optimization problem in terms of the natural parameters $\boldsymbol{m} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ and $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ is solved by minimizing the Lagrangian dual function, which is given as

$$\begin{aligned} g(\eta, \omega) = &\eta\epsilon - \omega\beta + \frac{1}{2}\big(\omega k \log(2\pi) - \eta(\log|\boldsymbol{\Lambda}_t^{-1}| + \boldsymbol{m}_t^{\mathrm{T}}\boldsymbol{\Lambda}_t^{-1}\boldsymbol{m}_t) \\ &+ (\eta + \omega)(\log|\boldsymbol{\Lambda}^{-1}| + \boldsymbol{m}^{\mathrm{T}}\boldsymbol{\Lambda}^{-1}\boldsymbol{m})\big), \end{aligned}$$

and the optimization problem becomes

$$\begin{aligned} \underset{\eta,\,\omega}{\text{minimize}} \quad & g(\eta, \omega) \\ \text{subject to} \quad & \eta \geq 0, \\ & \omega \geq 0 \end{aligned}$$

with Lagrangian multipliers η and ω. The convex dual function can be optimized using a constrained non-linear optimization method such as L-BFGS (Liu and Nocedal, 1989) to obtain the optimal solutions η^* and ω^*. The update rules for the new search distribution based on the Lagrangian multipliers are given by

$$\boldsymbol{\Lambda} = \frac{\eta^*\boldsymbol{\Lambda}_t + \boldsymbol{A}}{\eta^* + \omega^*}, \qquad \boldsymbol{m} = \frac{\eta^*\boldsymbol{m}_t + \boldsymbol{a}}{\eta^* + \omega^*}.$$

Mean and covariance can be recovered by the inverse transformations and are given by $\boldsymbol{\mu} = \boldsymbol{\Lambda}^{-1}\boldsymbol{m}$ and $\boldsymbol{\Sigma} = \boldsymbol{\Lambda}^{-1}$. We can now see that the new search distribution's parameters are an interpolation between the natural parameters of the old distribution and the parameters of the quadratic model from Equation (3).

3.2.1 Model Learning

The parameters of the quadratic model can be learned from the noisy evaluations $f(\boldsymbol{x})$ using linear regression with quadratic features. The original MORE algorithm first uses a probabilistic dimensionality reduction technique to project the problem parameters into a lower dimensional space. Afterwards, weighted Bayesian linear regression is used to solve for the model parameters in the reduced space. Lastly, the model parameters are projected back into the original space. This approach has several drawbacks, as it introduces additional hyper-parameters to the algorithm and involves a sample-based approach to integrate out the projection matrix which is very costly in terms of computation time. We will later on show that by using appropriate data pre-processing techniques, standard linear least squares is better suited to efficiently fit the quadratic models and also allows a direct connection of MORE to natural gradients.

3.2.2 Entropy Control

The entropy of the search distribution can be controlled with parameter β. The bound β is chosen such that entropy of the search distribution $H(\pi)$ decreases by a certain percentage until a minimum entropy H^0 is reached, i.e. $\beta = \gamma(H(\pi_t) - H^0) + H^0$. Alternatively, a simple alternative is to linearly decrease β in each iteration until the lower bound H^0 is reached, i.e. $\beta = \max(H(\pi_t) - \delta, H^0)$.

3.3 Relation to Natural Gradient

The natural gradient is an optimization technique that considers the geometry of the parameter space to improve the convergence speed and efficiency of training machine learning models, especially in situations where parameters are highly correlated (Amari, 1998). The natural gradient is given by scaling the "vanilla" gradient

$$\nabla_\theta J = \mathbb{E}_{\boldsymbol{x}\sim\pi}[\nabla_\theta \log \pi(\boldsymbol{x}) \mathcal{F}(\boldsymbol{x})] \approx \sum_i \nabla_\theta \log \pi(\boldsymbol{x}_i) f(\boldsymbol{x}_i)$$

with the inverse of the Fisher information matrix (FIM) $\boldsymbol{F} = \mathbb{E}_{\boldsymbol{x}\sim\pi}[\nabla_\theta \log \pi(\boldsymbol{x}) \nabla_\theta \log \pi(\boldsymbol{x})^{\mathrm{T}}]$, i.e.,

$$\boldsymbol{g}_{\mathrm{NG}} = \boldsymbol{F}^{-1} \nabla_\theta J.$$

3.3.1 Approximating the Natural Gradient with Samples

A sample based approximation of the natural gradient is given as

$$\begin{aligned}\boldsymbol{g}_{\mathrm{NG}} &\approx \left(\sum_i \nabla_\theta \log \pi(\boldsymbol{x}_i) \nabla_\theta \log \pi(\boldsymbol{x}_i)^{\mathrm{T}}\right)^{-1} \sum_i \nabla_\theta \log \pi(\boldsymbol{x}_i) f(\boldsymbol{x}_i) \\ &= \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{y} \end{aligned} \tag{4}$$

with

$$\boldsymbol{\Phi} = \begin{bmatrix} \nabla_\theta \log \pi(\boldsymbol{x}_1) \\ \vdots \\ \nabla_\theta \log \pi(\boldsymbol{x}_N) \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

where $y_i = f(\boldsymbol{x}_i)$, $i = 1, \dots, N$. Equation (4) resembles the least squares solution to a linear regression problem which is given by

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_i \left(\nabla_\theta \log \pi(\boldsymbol{x}_i)^{\mathrm{T}} \boldsymbol{w} - y_i\right)^2 \tag{5}$$

using the function approximator $\nabla_\theta \log \pi(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{w}$ where the features are given by the gradient of the log search distribution with respect to the parameters of π. In this case, the natural gradient is given by the least squares solution, i.e., $\boldsymbol{g}_{\mathrm{NG}} = \boldsymbol{w}^*$.

3.3.2 Compatible Function Approximation for Stochastic Search

Equation (5) is a special case of compatible function approximation. To see this, we will quickly review this concept. Originally defined in the context of policy gradients with function approximation for step-based reinforcement learning (Sutton et al., 1999), the compatible function theorem states that the policy gradient with function approximation is exact under two conditions:

1. The gradient of the advantage function approximator is equal to the log gradients of the policy, i.e. $\nabla_w \hat{A}(s, a; w) = \nabla_\theta \log \pi(a \mid s; \theta)$
2. the parameters w of the advantage function minimize a mean-squared error.

Stochastic search can be seen as a state-less one-step RL problem where $A(s, a) = \mathcal{F}(\boldsymbol{x})$ and the policy $\pi(a \mid s; \theta)$ is the search distribution $\pi(\boldsymbol{x})$. For the gradient to be unbiased in the case of MORE where we optimize an approximated surrogate objective, we need to replace $\mathcal{F}(\boldsymbol{x})$ with a learned approximation $\hat{\mathcal{F}}$ that uses compatible features. For a Gaussian distribution with natural parameters $\boldsymbol{m} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ and $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$, these features are given by

$$\begin{aligned}\nabla_{\boldsymbol{m}} \log \pi(\boldsymbol{x}) &= \boldsymbol{x}^{\mathrm{T}} + c_1, \\ \nabla_{\boldsymbol{\Lambda}} \log \pi(\boldsymbol{x}) &= -\frac{1}{2}\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}} + c_2.\end{aligned}$$

Thus, the compatible features are given by

$$\begin{aligned}\phi(\boldsymbol{x}) &= \left[\nabla_{\boldsymbol{m}} \log \pi(\boldsymbol{x}), \mathrm{vec}(\nabla_{\boldsymbol{\Lambda}} \log \pi(\boldsymbol{x}))\right] \\ &= \left[1, \boldsymbol{x}, -\tfrac{1}{2}\,\mathrm{vec}(\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}})\right].\end{aligned}$$

Consequently, if the model is obtained by a linear least squares fit using quadratic features, the MORE algorithm is performing an unbiased update in the natural gradient direction where the learning rate is specified implicitly by the trust region ϵ. This is reflected in the update equations of MORE which, assuming an entropy bound $\beta \to -\infty$, are simply given by

$$\boldsymbol{m} = \boldsymbol{m}_t + \eta^{-1}\boldsymbol{a}, \quad \boldsymbol{\Lambda} = \boldsymbol{\Lambda}_t + \eta^{-1}\boldsymbol{A},$$

where $\boldsymbol{a}$ and $\boldsymbol{A}$ are the estimated parameters of the compatible function approximation. They directly correspond to the fitted model parameter $\boldsymbol{w}^*$ in Equation (5). In difference to other methods such as CMA-ES (Akimoto et al., 2010) or NES, where a relation to natural gradients has also been established, the natural gradient update of MORE is exact since the estimation of the quadratic model is unbiased in expectation and no ranking based reward transformation is used.

4. Improving the MORE Algorithm

In its original formulation, MORE has several drawbacks as already pointed out earlier. In this section, we will introduce a new version of MORE that aims at improving convergence speed and simplifying the model learning approach. We achieve this by (a) disentangling the trust regions for mean and covariance, (b) an adaptive entropy control mechanism, (c) a simplified but improved model learning approach based on standard linear least squares. We also propose a robust fitting of the surrogate to stabilize the model fitting process. We will refer to the new algorithm as **C**oordinate-**A**scent MORE with **S**tep Size Adaptation or CAS-MORE for short.

4.1 Disentangled Trust Regions

The trust-region radius ϵ controls the change of the distributions between subsequent updates and thus has an influence on the speed of convergence. While a large trust region should encourage larger steps of the mean, we observed in our experiments that the result is mainly a large change in the covariance matrix leading to instabilities and divergence of the algorithm, especially on problems where it is difficult to estimate the quadratic model.

Therefore, limiting the KL-divergence between the policies in subsequent iterations is a sub-optimal choice as we have no direct control whether the mean or covariance of the search distribution is updated more aggressively. To alleviate this problem, we propose to decouple the mean and covariance update by employing a block coordinate ascent strategy for the mean and the covariance matrix which allows for setting different bounds on each of the components. This approach results in independent updates for the mean and covariance as in CMA-ES, albeit more principled, and has already been explored successfully in other episodic (Otto et al., 2023) and step-based (Abdolmaleki et al., 2018a) reinforcement algorithms.

The optimization problems we will be solving are

$$\begin{aligned} \underset{\boldsymbol{\mu}}{\text{maximize}} \quad & \int_{\boldsymbol{x}} \pi(\boldsymbol{x}) \hat{\mathcal{F}}(\boldsymbol{x}) \mathrm{dx} \Big|_{\boldsymbol{\Sigma}=\boldsymbol{\Sigma}_t} \\ \text{subject to} \quad & \mathrm{KL}(\pi(\boldsymbol{x}) \parallel \pi_t(\boldsymbol{x}))|_{\boldsymbol{\Sigma}=\boldsymbol{\Sigma}_t} \leq \epsilon_\mu \end{aligned} \tag{6}$$

for the mean and

$$\begin{aligned} \underset{\boldsymbol{\Sigma}}{\text{maximize}} \quad & \int_{\boldsymbol{x}} \pi(\boldsymbol{x}) \hat{\mathcal{F}}(\boldsymbol{x}) \mathrm{dx} \Big|_{\boldsymbol{\mu}=\boldsymbol{\mu}_t} \\ \text{subject to} \quad & \mathrm{KL}(\pi \parallel \pi_t)|_{\boldsymbol{\mu}=\boldsymbol{\mu}_t} \leq \epsilon_\Sigma \end{aligned} \tag{7}$$

for the covariance in each iteration. By choosing a small bound ϵ_Σ, we can drop the entropy constraint from the optimization as the entropy is not decreasing as quickly.

4.1.1 Updating the Mean

We start updating the mean by setting $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_t$ and introducing a bound ϵ_μ to limit the change of the mean displacement. The dual function to the problem in Equation (6) is given by

$$g_\mu(\lambda) = \lambda\epsilon_\mu + \frac{1}{2}\Big(\boldsymbol{m}_\mu(\lambda)^{\mathrm{T}} \boldsymbol{M}_\mu(\lambda)^{-1} \boldsymbol{m}_\mu(\lambda) - \lambda \boldsymbol{m}_t^{\mathrm{T}} \boldsymbol{M}_t^{-1} \boldsymbol{m}_t\Big)$$

where λ is a Lagrangian multiplier and

$$\boldsymbol{M}_\mu(\lambda) = \lambda\boldsymbol{\Sigma}_t^{-1} + \boldsymbol{A}, \qquad \boldsymbol{m}_\mu(\lambda) = \lambda\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\mu}_t + \boldsymbol{a}.$$

A detailed derivation of the dual function can be found in Appendix A.1. We now solve the dual problem

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \quad & g_\mu(\lambda) \\ \text{subject to} \quad & \lambda > 0 \end{aligned}$$

and obtain the new mean as

$$\boldsymbol{\mu}^* = \boldsymbol{M}_\mu(\lambda^*)^{-1}\boldsymbol{m}_\mu(\lambda^*)$$

where λ^* is the solution of the optimization problem which we find using a non-linear optimization algorithm [2].

4.1.2 Updating the Covariance Matrix

Next, we set $\boldsymbol{\mu} = \boldsymbol{\mu}_t$ and introduce a bound ϵ_Σ to constrain the change of the covariance. The dual function (see Appendix A.2 for a detailed derivation) for the problem in Equation (7) is given by

$$g_\Sigma(\nu) = \nu\epsilon_\Sigma + \frac{1}{2}\nu\Big(\log|\boldsymbol{\Lambda}(\nu)^{-1}| - \log|\boldsymbol{\Lambda}_t^{-1}|\Big)$$

with

$$\boldsymbol{\Lambda}(\nu) = \frac{\nu\boldsymbol{\Sigma}_t^{-1} + \boldsymbol{A}}{\nu}$$

and we need to solve the following optimization problem

$$\begin{aligned} \underset{\nu,\,\omega}{\text{minimize}} \quad & g_\Sigma(\nu) \\ \text{subject to} \quad & \nu > 0 \end{aligned}$$

where ν is again a Lagrangian multiplier. The optimal solution $\boldsymbol{\Sigma}^*$ in terms of the solution ν^* can be found analogously and is given by

$$\boldsymbol{\Sigma}^* = \boldsymbol{\Lambda}(\nu^*)^{-1}.$$

4.2 Entropy Control

The standard MORE algorithm is only able to decrease the entropy of the search distribution in each iteration. This might result in slow convergence in particularly if the search distribution is initialized with too small variances[3]. On the other hand, a too large covariance

2. We used the implementation of L-BFGS-B (Liu and Nocedal, 1989) provided by the python package NLOpt (Johnson, 2014).
3. This is often necessary in case a large initial exploration is generating samples that are very costly or dangerous to evaluate, e.g., in robotics.

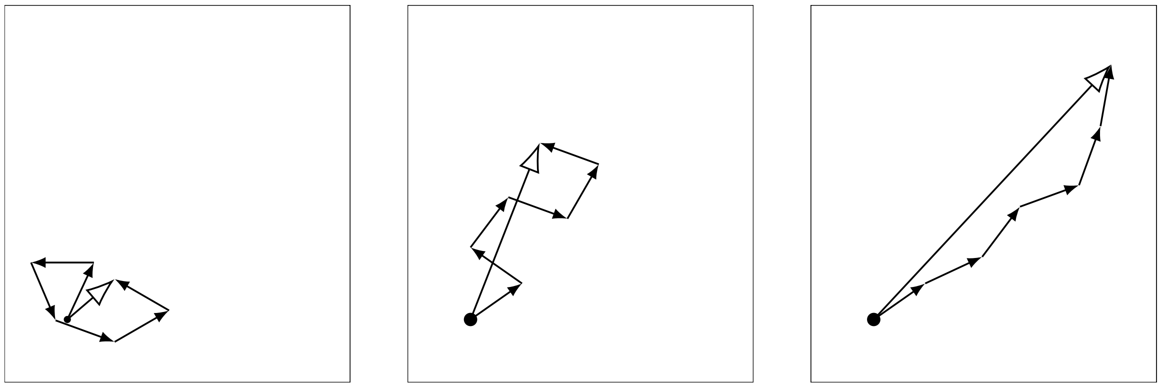


Figure 1: This figure illustrates the concept of the evolution path in a 2D example and is adapted from Hansen (2016). In each figure, arrows with black heads correspond to an update of the mean. The evolution path is a smoothed sum over subsequent mean updates and depicted with an open head. In the left plot, the mean updates show no clear search direction and, as a result, the evolution path is short. The right plot shows the opposite where heavily correlated mean updates lead to a long evolution path. The center plot shows the desired case, where subsequent mean updates are uncorrelated.

can also lead to slow convergence as it takes a long time to settle in on a solution and simply increasing the bound on the covariance can lead to unstable updates.

To solve these issues, we take inspiration from CMA-ES which already makes use of an entropy control mechanism in form of the step-size update (Hansen, 2016). The crucial property underlying this update is the so-called evolution path which is a smoothed sum over previous mean updates. The idea of step-size adaptation is the following. Whenever we take steps in the same direction, we could essentially take a single, larger step, while dithering around a constant location indicates no clear search direction. In the first case, a larger entropy would allow for bigger steps, in the second case, we need to decrease entropy. We illustrate the concept in Figure 1. We can achieve this by scaling the covariance after the optimization with a scalar factor

$$\sigma_{t+1} = \exp\left(-\frac{\delta_{t+1}}{n}\right)$$

after the optimization steps where n is the problem dimensionality and δ_{t+1} is the desired change between the entropy in iteration t and the next iteration $t+1$ of the algorithm. The question is now how to determine a suitable value for δ_{t+1}.

The step size update we propose is inspired by the step size control mechanism used in CMA-ES called cumulative step size adaptation (CSA). We also use the evolution path $\boldsymbol{p}$ which is a vector tracking previous mean updates and compare its length to a hypothetical length of the evolution path resulting from uncorrelated mean updates. Therefore, the evolution path update needs to be constructed in a way that we can derive a computable desired length from it. Inspired by CMA-ES, we initialize the evolution path $\boldsymbol{p}_0$ to zero and update it as follows

$$\boldsymbol{p}_{t+1} = (1 - c_\sigma)\boldsymbol{p}_t + \sqrt{\frac{c_\sigma(2 - c_\sigma)}{2\epsilon_\mu}}\boldsymbol{\Sigma}_t^{-\frac{1}{2}}(\boldsymbol{\mu}_{t+1} - \boldsymbol{\mu}_t)$$

where $c_\sigma < 1$ is a hyper parameter. The factors are chosen analogously to CMA-ES such that $(1 - c_\sigma)^2 + \sqrt{c_\sigma(2 - c_\sigma)}^2 = 1$. While multiplying the shift in means with the inverse square root of the covariance matrix in CMA-ES makes the evolution path roughly zero mean and unit variance distributed, we know its exact length due to the constraint on the mean update in Equation (6) as long as the mean update is at its bound. The vector $\boldsymbol{\Sigma}_t^{-\frac{1}{2}}(\boldsymbol{\mu}_{t+1} - \boldsymbol{\mu}_t)$ has in that case length $\sqrt{2\epsilon_\mu}$ and by scaling it with $(\sqrt{2\epsilon_\mu})^{-1}$ the resulting length is 1. We set

$$\delta_{t+1} = \alpha\left(1 - \frac{\|\boldsymbol{p}_{t+1}\|}{\|\boldsymbol{p}_{t+1}^{\text{des}}\|}\right),$$

where $\|\boldsymbol{p}_{t+1}^{\text{des}}\|$ is a desired length of the evolution path given by the length of a evolution path resulting from fully uncorrelated updates and α influences the magnitude of entropy change which was set to 1 in the black-box optimization experiments, and 0.1 in the episodic RL experiments. This rule for determining the change in entropy is almost identical to the one in CMA-ES. If the norm of the current evolution path $\boldsymbol{p}_{t+1}$ is smaller than $\boldsymbol{p}_{t+1}^{\text{des}}$, i.e., the mean updates were dithering around a constant location, we reduce entropy (i.e. $\delta_{t+1} > 0$) while entropy is increased if the length of the evolution path is longer than desired.

In order to determine the length of an uncorrelated path $\|\boldsymbol{p}_{t+1}^{\text{des}}\|$, we first look at the length of two vectors

$$\|\boldsymbol{a} + \boldsymbol{b}\| = \sqrt{\|\boldsymbol{a}\|^2 + \|\boldsymbol{b}\|^2 + 2\|\boldsymbol{a}\|\|\boldsymbol{b}\|\cos\theta},$$

where θ is the angle between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. If these two vectors are uncorrelated (i.e. $\theta = \pi/2$), the length becomes $\sqrt{\|\boldsymbol{a}\|^2 + \|\boldsymbol{b}\|^2}$. Under the assumption that the mean update is always at the bound, i.e. $(2\epsilon_\mu\boldsymbol{\Sigma}_t)^{-\frac{1}{2}}(\boldsymbol{\mu}_{t+1} - \boldsymbol{\mu}_t) = 1$, the length of the evolution path p_{t+1} given the previous length p_t is then

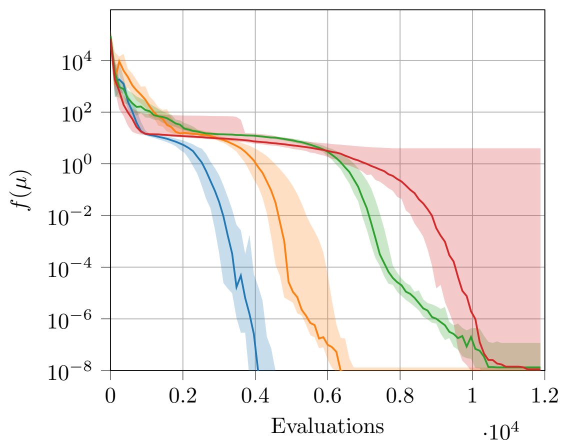
$$p_{t+1} = \sqrt{(1 - c_\sigma)^2 p_t^2 + c_\sigma(2 - c_\sigma)}.$$

In practice, we set the desired angle $\theta = \frac{3\pi}{8}$ and compute $\|\boldsymbol{p}_{t+1}^{\text{des}}\|$ accordingly to account for unavoidable correlations between iterations due to sample reuse and constrained updates. Finally, the adapted policy is given by

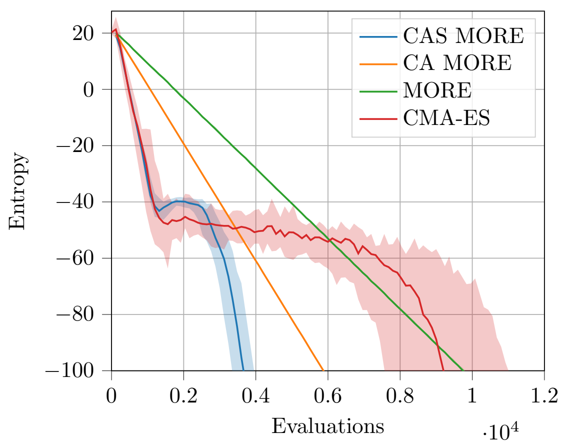
$$\pi_{t+1}^{\sigma} = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_{t+1}, \sigma_{t+1}^2\boldsymbol{\Sigma}_{t+1}).$$

4.3 Illustrative Example

We demonstrate the characteristics of CAS-MORE by comparing it to the original formulation of MORE and Coordinate Ascent MORE without adaptive entropy control (CA-MORE). We therefore use a 15 dimensional Rosenbrock function as defined in Hansen et al. (2009b) and run each variant of MORE 20 times with different seeds. The optimization is stopped once



(a) Function value evaluated at μ_t



(b) Entropy of the search distribution

Figure 2: This figure shows the function value (left) and the entropy of the search distribution (right) over the course of optimization of a 15-dimensional Rosenbrock function using different variants of MORE. The original MORE is shown in green, while coordinate ascent versions of more are orange and blue where CA-MORE indicates a fixed entropy reduction schedule and CAS-MORE indicates the adaptive entropy schedule using the step-size adaptation. For comparison, CMA-ES is plotted in red.

a target function value of 1×10^{-8} is reached, or 12 000 function evaluations are exceeded. Figure 2 shows median and 5% / 95% quantiles of the function value at the mean (Figure 2a) and the entropy of the search distribution (Figure 2b) over the optimization. The hyper-parameters for the KL bounds (and entropy schedule for MORE) are chosen individually for each algorithm based on a grid search. We observe that decoupling the mean and covariance update already significantly improves convergence speed as it allows for a higher learning rate for the mean and therefore a quicker entropy reduction. Note, that reducing entropy even quicker leads to premature convergence while increasing ϵ for MORE leads to divergence of the algorithm. This problem is alleviated by introducing the adaptive entropy schedule that first quickly reduces entropy, then plateaus, and, finally, quickly reduces entropy again, resulting in an accelerated optimization process.

5. Model Learning

In this section, we introduce a simple but effective approach of learning a quadratic model based on polynomial ridge regression using the method of least squares as replacement for the more complex Bayesian dimensionality reduction technique introduced in the original paper (Abdolmaleki et al., 2015). We design the model learning with two objectives in mind. First, it should be data- and time-efficient to allow for quick execution of the algorithm. Additionally, it should be robust towards outliers and samples from very low entropy regimes. To this end, we apply a series of data pre-processing techniques, re-use old samples, and start learning a model with a lower complexity and increase it once sufficient data is available.

5.1 Least Squares Model Fitting

In each iteration, we generate a fixed number K of samples $\boldsymbol{x}_i \sim \pi$, $i = 1, \dots, K$, evaluate them on the objective function to obtain $y_i = f(\boldsymbol{x}_i)$ and add the tuple $(\boldsymbol{x}_i, y_i)$ to a data-set $\mathcal{D} = \{(\boldsymbol{x}_q, y_q) \mid q = 1 \dots Q\}$ of length Q[4]. The goal of the model fitting process is to find the parameters $\boldsymbol{A}$, $\boldsymbol{a}$, and a of a quadratic model of the form

$$\mathcal{F}(\boldsymbol{x}) \approx \hat{\mathcal{F}}(\boldsymbol{x}) = -1/2\, \boldsymbol{x}^{\mathrm{T}} \boldsymbol{A} \boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}} \boldsymbol{a} + a.$$

These parameters can be found by solving a regularized least squares problem

$$\min_{\boldsymbol{w}} \|(\boldsymbol{y} - \boldsymbol{\Phi} \boldsymbol{w})\|_2^2 + \lambda \|\boldsymbol{w}\|_2^2,$$

where $\boldsymbol{\Phi}$ is the design matrix whose rows are given by a feature transformation $\phi(\boldsymbol{x})$, i.e $\boldsymbol{\Phi} = [\phi(\boldsymbol{x}_1), \dots, \phi(\boldsymbol{x}_Q)]^{\mathrm{T}}$, $\boldsymbol{y}$ is a vector containing the fitness evaluations $y_i = f(\boldsymbol{x}_i)$ and λ is the regularization factor. The solution is given by the well known ridge regression estimator

$$\hat{\boldsymbol{w}} = (\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{y}.$$

5.2 Adaptive Model Complexity

A full quadratic model has in the order of $\mathcal{O}(n^2)$ parameters that need to be estimated. Compared to model-free algorithms, this can be a disadvantage if we initially need to sample enough parameters for the first model to be built. Instead, the complexity of the feature function $\phi(\boldsymbol{x})$ will be gradually increased from a linear to a diagonal and, finally, a full quadratic model depending on the number of obtained samples. The simplest model is a linear model where

$$a = w_1 \qquad \boldsymbol{a} = [w_2, \dots, w_{n+1}]^{\mathrm{T}} \qquad \boldsymbol{A} = \boldsymbol{0}.$$

Next, we estimate a diagonal model where

$$\boldsymbol{A} = -\mathrm{diag}([w_{n+2}, \dots, w_{2n+1}]).$$

Once sufficiently many data-points are available, we estimate a full quadratic model where

$$\boldsymbol{A} = -(\boldsymbol{L} + \boldsymbol{L}^{\mathrm{T}})$$

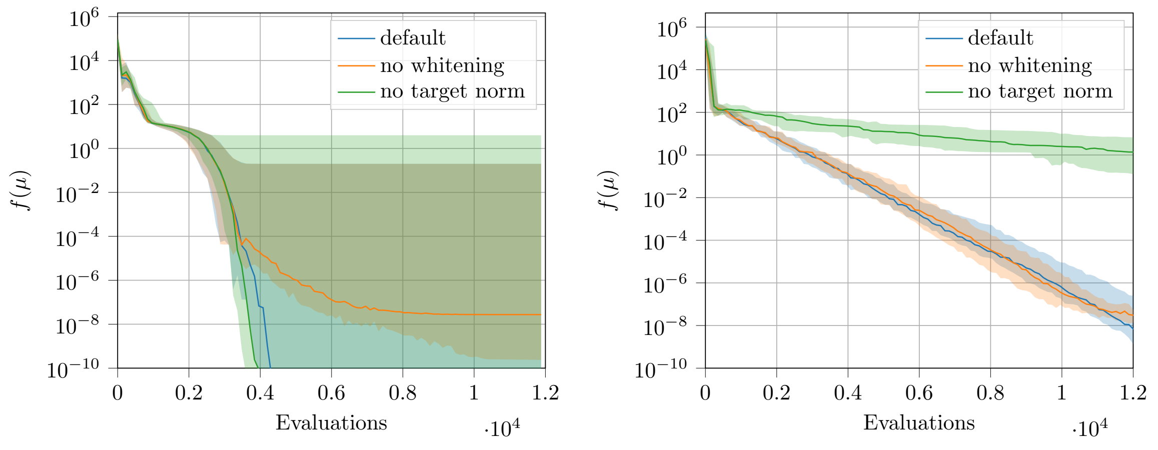
with

$$\boldsymbol{L} = \begin{bmatrix} w_{n+2} & 0 & \cdots & \cdots & 0 \\ w_{n+3} & w_{n+4} & 0 & \cdots & 0 \\ \vdots & & \ddots & & \\ w_{n(n+3)/2-1} & \cdots & \cdots & w_{n(n+3)/2-n-1} & 0 \\ w_{n(n+3)/2-n+2} & \cdots & \cdots & \cdots & w_{n(n+3)/2+1} \end{bmatrix}.$$

The feature functions for the models are given by

$$\begin{aligned} \phi_{\mathrm{lin}}(\boldsymbol{x}) &= [1, x_1, x_2, \dots, x_n]^{\mathrm{T}} \\ \phi_{\mathrm{diag}}(\boldsymbol{x}) &= [\phi_{\mathrm{lin}}(\boldsymbol{x})^{\mathrm{T}}, x_1^2, x_2^2, \dots, x_n^2]^{\mathrm{T}} \\ \phi_{\mathrm{full}}(\boldsymbol{x}) &= [\phi_{\mathrm{lin}}(\boldsymbol{x})^{\mathrm{T}}, x_1^2, x_1 x_2, x_1 x_3, \dots, x_1 x_n, x_2^2, x_2 x_3, \dots x_2 x_n, x_3 x_4, \dots, x_n^2]^{\mathrm{T}}. \end{aligned}$$

4. Once the number of data-points Q in $\mathcal{D}$ exceeds a maximum queue-size $Q_{\max}$, we discard old samples in a first-in, first-out manner.



(a) Rosenbrock (b) Attractive Sector

Figure 3: This figure shows the effects of data pre-processing on the optimization. We plot the function value over the course of optimization for a Rosenbrock (a) and Attractive Sector (b) function in 15 dimensions and turn off whitening and robust target normalization with clipping, respectively. We see that whitening becomes important for low entropy regimes near the optimum, while target normalization is especially important in case of outliers in the function values as can be seen for the Attractive Sector function.

We empirically found that we require at least 10% more samples than the model has parameters, i.e. start with a linear model once $|\mathcal{D}| \geq 1.1(1+n)$ and continue with a diagonal model once $|\mathcal{D}| \geq 1.1(1+2n)$. Finally, we switch to estimating a full quadratic model once $|\mathcal{D}| \geq 1.1(1+n(n+3)/2)$.

5.3 Data Pre-Processing

Estimating the model parameters can be numerically difficult with function values spanning several orders of magnitude, outliers, and very low variances of input and output values near the optimum. Therefore, we perform data pre-processing on the input values $\boldsymbol{x}$, as well as the design matrix $\boldsymbol{\Phi}$ and the target values y before solving the least squares problem. In particular, we perform whitening of the inputs $\boldsymbol{x}$, standardization of the design matrix $\boldsymbol{\Phi}$ and a special form of standardization of the target values y and construct the normalized data set $\mathcal{D}_w$. We demonstrate the effect of data pre-processing in Figure 3.

5.3.1 Data Whitening

Before estimating $\hat{\boldsymbol{\beta}}$, we whiten the input data and obtain

$$\boldsymbol{x}_w = \bar{\boldsymbol{C}}_{\mathcal{D}}^{-1}(\boldsymbol{x} - \bar{\boldsymbol{x}}_{\mathcal{D}}) \; \forall \; \boldsymbol{x} \in \mathcal{D}$$

where $\bar{\boldsymbol{x}}_{\mathcal{D}}$ is the empirical mean and $\bar{\boldsymbol{C}}_{\mathcal{D}}$ is the Cholesky-factorization of the empirical covariance of the data-set. The parameters $\hat{\boldsymbol{\beta}}_w$ learned with the whitened data-set to be transformed back into unwhitened parameters. This transformation is given by

$$\begin{aligned}
\boldsymbol{A} &= \bar{\boldsymbol{C}}_{\mathcal{D}}^{-\mathrm{T}} \boldsymbol{A}_w \bar{\boldsymbol{C}}_{\mathcal{D}}^{-1}, \\
\boldsymbol{a} &= \boldsymbol{A}\bar{\boldsymbol{x}}_{\mathcal{D}} + \bar{\boldsymbol{C}}_{\mathcal{D}}^{-\mathrm{T}} \boldsymbol{a}_w, \\
a &= a_w + \bar{\boldsymbol{x}}_{\mathcal{D}}^{\mathrm{T}} (\boldsymbol{A}\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{C}}_{\mathcal{D}}^{-\mathrm{T}} \boldsymbol{a}_w),
\end{aligned}$$

where $\boldsymbol{A}_w$, $\boldsymbol{a}_w$ and a_w are the model parameters in the whitened space.

5.3.2 Target Normalization

Normalization of function values is beneficial as it stabilizes the model learning process. Depending on the number of samples we draw in each iteration as well as how well behaved the objective function is, we propose two target normalization methods. The first one allows for exact natural gradient updates, while the second one is more robust towards outliers.

Standard Target Normalization The first target normalization method is based on a simple standardization

$$y_w = (y - \bar{y}_{\mathcal{D}})/s_{\mathcal{D}} \; \forall \; y \in \mathcal{D}$$

of the target values where $\bar{y}_{\mathcal{D}}$ is the empirical mean and $s_{\mathcal{D}}$ is the empirical standard deviation of the target values in $\mathcal{D}$.

Robust Target Normalization Additionally, we propose an adaptive iterative normalization and clipping scheme based on the excess kurtosis of the target values resulting in a normalization process more robust towards outliers. Given the data-set in a particular iteration, we first perform standardization like above, then treat all values outside the interval $[-v_{\text{clip}}, v_{\text{clip}}]$ as outliers. Afterwards, we look at the excess kurtosis of the standardized targets in the interval $(-v_{\text{clip}}, v_{\text{clip}})$. If it is high, the data is still compacted to a small interval while outliers at the borders negatively influence the least squares optimization. In order to evenly spread the data, we repeat the procedure (normalization and clipping), until the excess kurtosis falls below a threshold or the standard deviation of the remaining values is close to 1. After the procedure, we replace the negative and positive outliers with the minimum and maximum value of the remaining targets, respectively. We illustrate the procedure in Figure 4, pseudo-code of the approach can be found in Appendix B. While this effectively changes the optimization objective, it is not as severe as replacing all function values with a ranking and, thus, the update direction still corresponds to an approximated natural gradient. An exact quantification of the error is subject to future work.

6. Experiments

In this section, we evaluate the performance of CAS-MORE[5] in the black-box stochastic search and episodic reinforcement learning setting. While we use the former to highlight the applicability over a wide range of objective functions using as few samples as possible, it is the episodic reinforcement learning domain where the core strengths of the algorithm come into play. We start by looking at the performance in terms of sample efficiency on a set of

5. An implementation of the algorithm can be found under `https://github.com/ALRhub/cas-more`

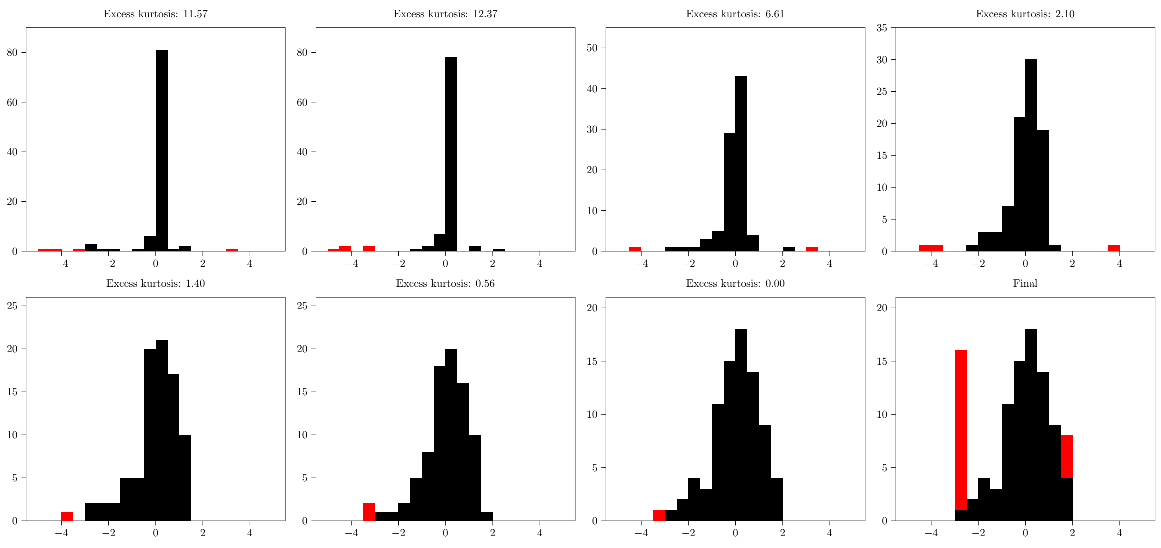


Figure 4: This figure illustrates the robust target normalization scheme on 100 synthetic data points, generated by a reward function $y = -0.5\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}$. We sample $\boldsymbol{x}$ from a two dimensional multivariate normal distribution with zero mean and unit variance and pick 20 random y values and add Gaussian noise with a standard deviation of 10. Plots from top left to bottom right show histograms of the data after standardization of the input in each level of recursion of the procedure. The excess kurtosis is measured on the black data points in the interval (-3, 3) and only these data points are recursively treated again until the excess kurtosis of the clipped data is below the threshold of 0.55. Finally, the bottom right histogram shows the output with the red data points clipped to the minimum and maximum values of the remaining data points. Note, that in each plot the standardized data has zero mean and unit variance but the model quality suffers from the present outliers.

black-box optimization benchmark functions provided by Hansen et al. (2021) and compare CAS-MORE to the original formulation of MORE, as well as competitors such as CMA-ES (Hansen, 2016) and XNES (Wierstra et al., 2014). Afterwards, we run our algorithm on a suite of different episodic reinforcement learning tasks from the domain of robotics where the fitness function evaluation is inherently noisy and we also care about the quality of the generated samples (e.g., the sample evaluations should not damage the robot) instead of plainly looking at the fitness evaluation at the mean of the search distribution. We compare against state of the art episodic and step-based reinforcement learning algorithms.

6.1 Black-Box Optimization Benchmarks

The COCO framework (Hansen et al., 2021) provides continuous benchmark functions to compare the performance of optimizers on problems with different problem dimensions. In these deterministic problems, it is only important to find a good point estimate $\boldsymbol{x}^*$. We choose this benchmark to show that our algorithm is competitive with other black-box optimizers in using as few samples as possible. In order to avoid the center-bias problem (Kudela, 2022), the optimum of the objective function is randomly shifted away from zero for each instance.

6.1.1 Experimental Setup

We evaluate MORE on the 24 functions of the Black-box Optimization Benchmark (BBOB) (Hansen et al., 2009a) suite in dimensions 2, 3, 5, 10, 20 and 40. We run the experiments without explicit optimization of the algorithm's hyper parameters for individual functions. Table 2 shows a set of default parameters which we found to robustly perform well on a wide variety of benchmark functions in terms of the problem dimensionality n. Additionally, we allow restarts of the algorithm with a larger population size whenever the optimization process fails (see (Auger and Hansen, 2005)) until a budget of $10\,000n$ function evaluations is exceeded or the final target of 1×10^{-8} is reached. Since MORE maximizes, we multiply the function by -1 in order to minimize the objective.

6.1.2 Black-box Optimization Benchmarks

Figure 5 shows runtime results aggregated over function groups and on single functions in dimension 20. Plotted is the percentage of targets reached within the interval 1×10^2 to 1×10^{-8} versus the log of objective function evaluations divided by the problem dimensionality. The further left a curve is, the quicker it reached a certain target. The first two rows correspond to combined results on separable, moderate, ill-conditioned, multi-modal and weakly structured multi-modal functions, as well as combined results from all 24 functions. The third and fourth row shows results on selected individual functions.

We first notice that CAS-MORE has a significant advantage over the original formulation of MORE on almost all functions. Furthermore, we can see that, especially on functions from the group of moderate and ill-conditioned functions such as Ellipsoid, Rosenbrock or Bent Cigar, CAS-MORE is able to improve state of the art results of competitors like CMA-ES and XNES. On other functions on the other hand, for example the Attractive Sector function, MORE has slower convergence. We found this shortcoming is mainly due to difficulties estimating a good model for this objective (see also Figure 3b). Other hard functions include multi-modal functions such as the Rastrigin function, where MORE often focuses on a local optimum. Here, the solution is to draw more samples in each iteration to improve the estimated model. For more results on the BBOB suite we refer to Appendix D. Typically, these black-box optimization benchmarks also focus purely on the fitness evaluation of the mean of the distribution and disregard the quality of the samples. While CAS-MORE is comparable to state-of-the-art black-box optimizers such as CMA-ES in these domains, the real benefits of MORE appear if we consider problems with noisy fitness evaluations and whenever we care also about the quality of the generated samples. This is for example the case in robot reinforcement learning problems which are discussed in the next sub-section.

6.2 Episodic Reinforcement Learning Results

Our experiments aim to showcase the benefits of CAS-MORE over other black-box optimization techniques in the episodic RL setting, as well as deep RL algorithms for the step-based RL setting. To this end, we compare CAS-MORE to the original MORE algorithm that uses the dimensionality reduction model, an intermediate algorithm labeled as MORE-LS comprising of the same joint KL and entropy constraint as in MORE and the simplified model learning approach as in CAS-MORE, and CMA-ES, representing black-box stochastic search optimizers. Furthermore, we compare to policy gradient based episodic reinforcement learning using movement primitives with and without trust regions (Otto et al., 2023) labeled as BBRL-TRPL and BBRL-PPO, respectively. Note that these algorithms have been developed for contextual episodic RL, which allows to learn a deep neural network mapping from a context vector, which describes the task, to a motion primitive parameter vector, which describes the corresponding robot motion. In this work, we concentrate on the non-contextual case where we only have to learn a single Gaussian distribution. While this case is arguably simpler, we show that CAS-MORE significantly outperforms trust-region policy gradient based methods, showing the benefits of a closed form natural gradient update. Finally, we compare with contemporary step-based deep RL algorithms such as PPO, SAC, and TD-3. We run step-based PPO using approximately the same number of transitions as the episode-based algorithms. For SAC and TD-3, we are limited by a 24 hour compute budget and therefore report the performance after this budget is exceeded.

In our experiments, we analyze the performance of the mean the search distribution, as well as other performance indicators such as success rates. The experiments are designed to be non-contextual with unchanged starting configurations of the agent in a noisy environment. For the episodic RL experiments, we use Probabilistic Movement Primitives (Paraschos et al., 2013) to plan trajectories. A PD-controller provides the torques necessary to follow them. For the step-based deep RL algorithms, we provide proprioceptive feedback in form of the joint angles and joint velocities, as well as a normalized time-step to the agent. The policy directly outputs the torques applied to the joints. For the BBRL and deep RL experiments, we orient ourselves towards the hyper-parameters used in Otto et al. (2023). They are provided in Appendix C. For more careful exploration, we set $\epsilon_\mu = 0.05$ and $\epsilon_\Sigma = 0.005$ and draw 64 samples per iteration for the episodic RL algorithms. Since we do not operate using a very low number of new samples per iteration and the reward functions do not produce large outliers, we can use the standard target normalization allowing for more exact natural

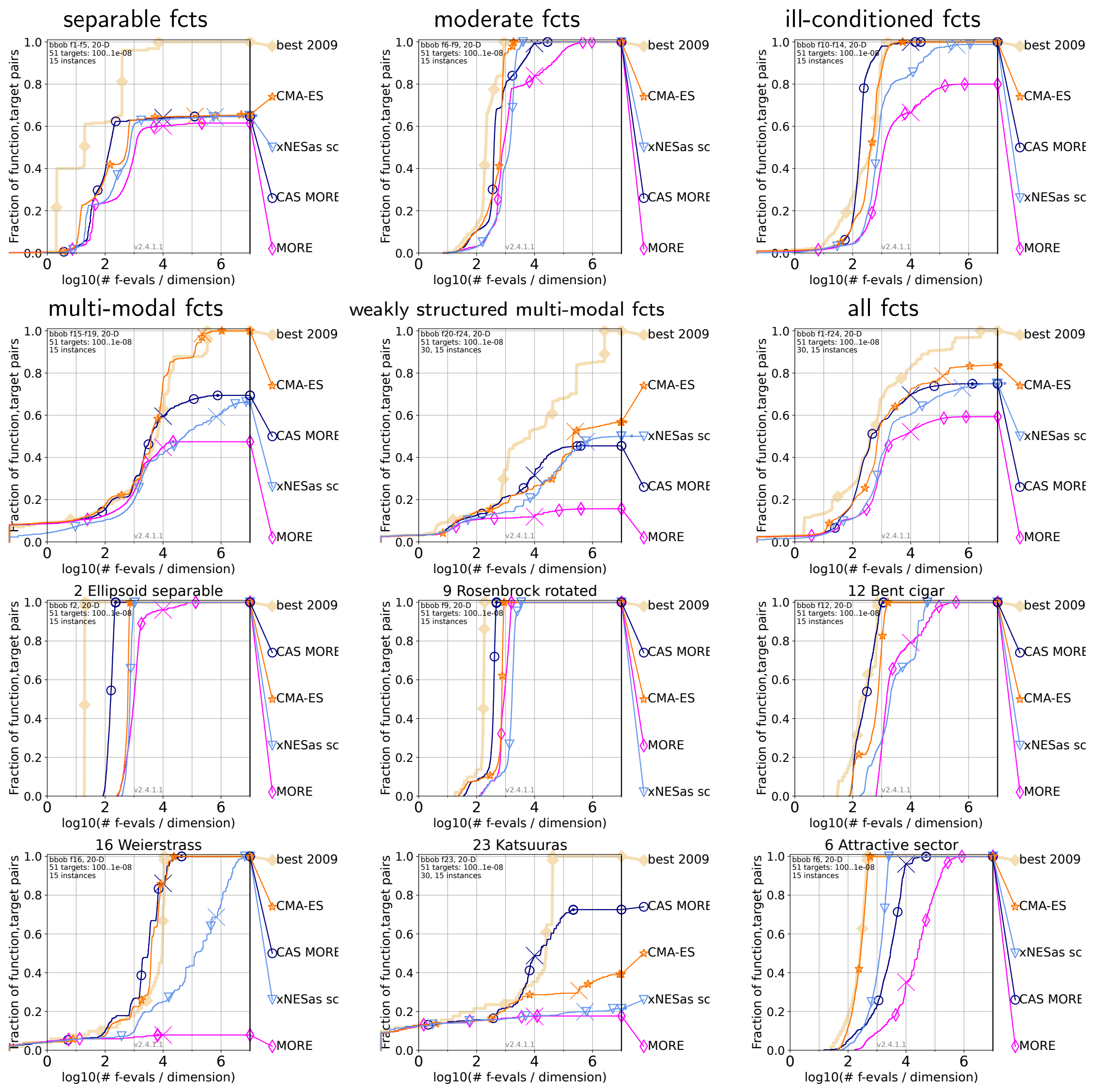


Figure 5: This figure shows the bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ in 20-D representing the percentage of targets achieved over the number of function evaluations. The results are averaged over 15 instances of each function. As reference algorithm, the best algorithm from BBOB 2009 is shown as light thick line with diamond markers. The first two rows show the aggregated results for all functions and subgroups of functions. Additionally, we show the results of individual functions in the third and fourth row. Big thin crosses indicate the used budget median for the respective algorithm which is 10 000 n function evaluations for each trial of CAS-MORE. Runtimes to the right of the cross are based on simulated restarts and are used to determine a runtime for unsuccessful runs (Hansen et al., 2016c).

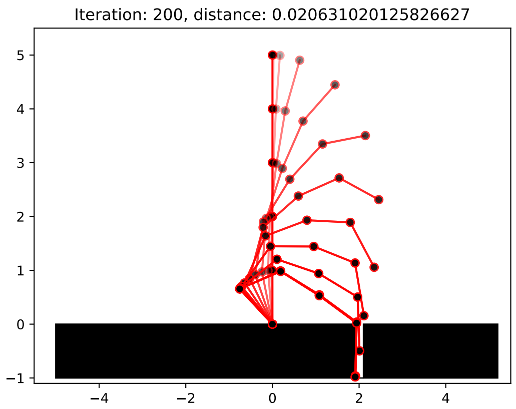


Figure 6: This figure shows an illustration of the hole reacher task. The robot arms starts upright and smoothly reaches down the narrow hole. The policy is learned using CAS-MORE and shows a typical behavior that keeps a safe distance to the ground.

gradient updates. We report the results in terms of the interquartile mean (IQM) with a 95% stratified bootstrap confidence interval (Agarwal et al., 2021).

6.2.1 Hole Reacher

The first task is an advanced version of the holereaching task from Abdolmaleki et al. (2015) that aims to show the difference between step-based and episodic reward formulations and the benefits of maximizing the expected fitness. The end-effector of a 5-link planar robot arm has to reach into a narrow hole without colliding with the ground or the walls of the hole. The arm is controlled in joint-space by applying torques to each of the 5 joints. Each link has a length of 1 m, the hole has a width of 20 cm, a depth of 1 m and is located 2 m away from the robot's base. For an illustration of a successful episode, see Figure 6. While the simulation of the arm is rather crude, this experiment has other properties that emphasize certain shortcomings of step-based RL, as well as rank-based optimization techniques. These properties include two distinct regions of exploration where above ground level, the agent can explore freely, while below the ground level even slight errors lead to a collision with the walls of the hole. Additionally, the depth of the hole, which is a main contributor to the reward function, is subject to noise in the noisy environment.

For the episodic RL setting, we learn the parameters of a ProMP with 3 basis functions, resulting in 15 parameters to be optimized. An episode consists of 200 time steps and in each iteration of the learning algorithm, we draw 64 new samples for the episodic RL algorithms. The cost function of the task is composed of two stages. First, the distance of the end-effector to the entrance of the hole is minimized, subsequently, the reward increases the further down the end-effector reaches. The reward is scaled down with a constant factor if the robot touches the ground. Additional penalty costs for acceleration ensure a smooth and energy-efficient trajectory. This reward function is different from the original MORE paper and is designed to highlight the risk awareness of the tested algorithms as the reward increases towards the bottom of the hole and then suddenly drops. The exact reward function can be found in Appendix E.1.

We examine 4 different settings of this experiment where we highlight the consequences of each choice on the learning process, namely

1. noise-free environment with dense in time reward,
2. noisy environment with dense in time reward,
3. noise-free environment with sparse in time reward,
4. noisy environment with sparse in time reward,

where in the noisy environment the perception of the depth of the hole is subject to noise[6]. The difficulty of this task lies in the two distinct sectors above and below the ground level. While above ground level, the agent can explore without negative consequences, it has to be much more careful once navigating into the narrow hole. The results of these experiments can be found in Figure 7. We plot the performance at the mean of the search distribution in left column, the expected performance under the search distribution in the center left column, the percentage of samples that collided with the ground in the center right column, and the distance of the end-effector to the bottom of the hole at the end of the episode in the right column. The result is averaged over 20 seeds for the episodic RL algorithms and 10 seeds for the step-based RL algorithms.

Dense in time reward We first examine the results of the experiment in the noise-free environment with dense in time reward, the usual setting of current deep RL literature, in the first row of Figure 7. When looking at the second column, we can see that CAS-MORE optimizes towards a stable solution reaching into the ground that avoids collision with the ground. While CMA-ES finds a solution that comes closer to the bottom of the hole (right column) leading to a higher performance at the mean, it often draws samples that collide with the walls during the optimization as can be seen in the center left column. Compared to the original MORE algorithm, MORE-LS using a model based on our newly introduced model learning approach leads to a similar performance as CAS-MORE. BBRL-TRPL also performs similarly to CAS-MORE, yet, at the cost of much higher computation time. For a comparison of run times, see also Table 1. PPO optimizes towards fast movement of the end-effector in the direction of the entrance of the hole but fails to consistently reach into it. When looking at individual learning curves, we found that it often finds policies that successfully reach into the hole but soon after forgets this solution again. SAC, TD-3, BBRL-PPO, and also the original MORE with the dimensionality reduction model fail to find policies that reach into the ground in our experiments.

Next, we look at the results of the experiments in the noisy environment in the second row of Figure 7. Compared to the noise-free experiment, we can see the biggest impact on CMA-ES. CMA-ES again optimizes very fast towards the ground of the hole but due to the noise it quickly ends up in a solution that collides with the ground. CAS-MORE, MORE-LS and BBRL-TRPL on the other hand are hardly influenced by the noise in the environment and still provide solutions that reach into hole, albeit maintaining a slightly larger distance

6. In the beginning of the episode, we sample the depth of the hole uniformly from $(1.00 \pm 0.02)\,\mathrm{m}$.

Algorithm	Median (s)	Mean (s)	Standard Deviation (s)
BBRL PPO	0.23196	0.25007	0.03200
BBRL TRPL	0.42725	0.43615	0.11009
CAS MORE	0.01095	0.01117	0.00191
CMA-ES	0.00601	0.00613	0.00192
MORE	2.24552	2.20081	0.36460
MORE LS	0.01070	0.01462	0.08261

Table 1: Runtime comparison for various episodic reinforcement learning algorithms. We measure the time the algorithm takes for one iteration without the sampling process.

to the ground compared to the noise-less case. The results of the rest of the algorithms stay more or less unchanged as they never get close to the ground.

Sparse in time reward As noted in Otto et al. (2023), a dense reward leads to a fast movement and overall poor energy efficiency. Thus, a better task description for this task is to only provide a distance-based reward in the last time-step and penalize high action values in each time-step. Again, we first examine the noise-free environment results which can be found in the third row of Figure 7. Even without noise, the results of CMA-ES become inconsistent over individual seeds, while CAS-MORE consistently optimizes towards well performing distributions. We assume this is the case as the distance between the initialization of the movement primitive parameters and a well performing solution is quite large. MORE-LS and BBRL-TRPL still find good solutions but perform slightly worse as can be seen from the distance of the end-effector to the ground in the last column. All step-based algorithms now fail to find good policies.

Last, we look at the results of the experiments in the noisy environment in the last row of Figure 7. We can now see the full potential of optimizing the expected reward and individual updates of the mean and covariance of the search distribution, as CAS-MORE finds the policies with the best performance at the mean as well as the best performing samples during the optimization process, with MORE-LS and BBRL-TPRL slightly below them. CMA-ES now consistently produces policies that collide with the ground. Results for all other algorithms remain poor.

6.2.2 Table Tennis

In the second task, we want to teach a 7 DoF Barrett WAM robotic arm a fore-hand smash. The goal is to return a table-tennis ball and place it as close as possible to the far edge of table on the opponent's side. For this and all following experiments, we concentrate on the noisy environment with sparse in time reward setting. Thus, in every episode, the ball is initialized at the same position but the velocity in x-direction (approaching the robot) is noisy. A robust strategy is to account for the uncertain velocity and aim for a spot that is not too close to the edge. An illustration of a successful execution of the task can be found in Figure 8.

We learn the parameters of a ProMP where we use 2 basis functions for each joint, resulting in 14 parameters to be optimized. The weights of the movement primitive are

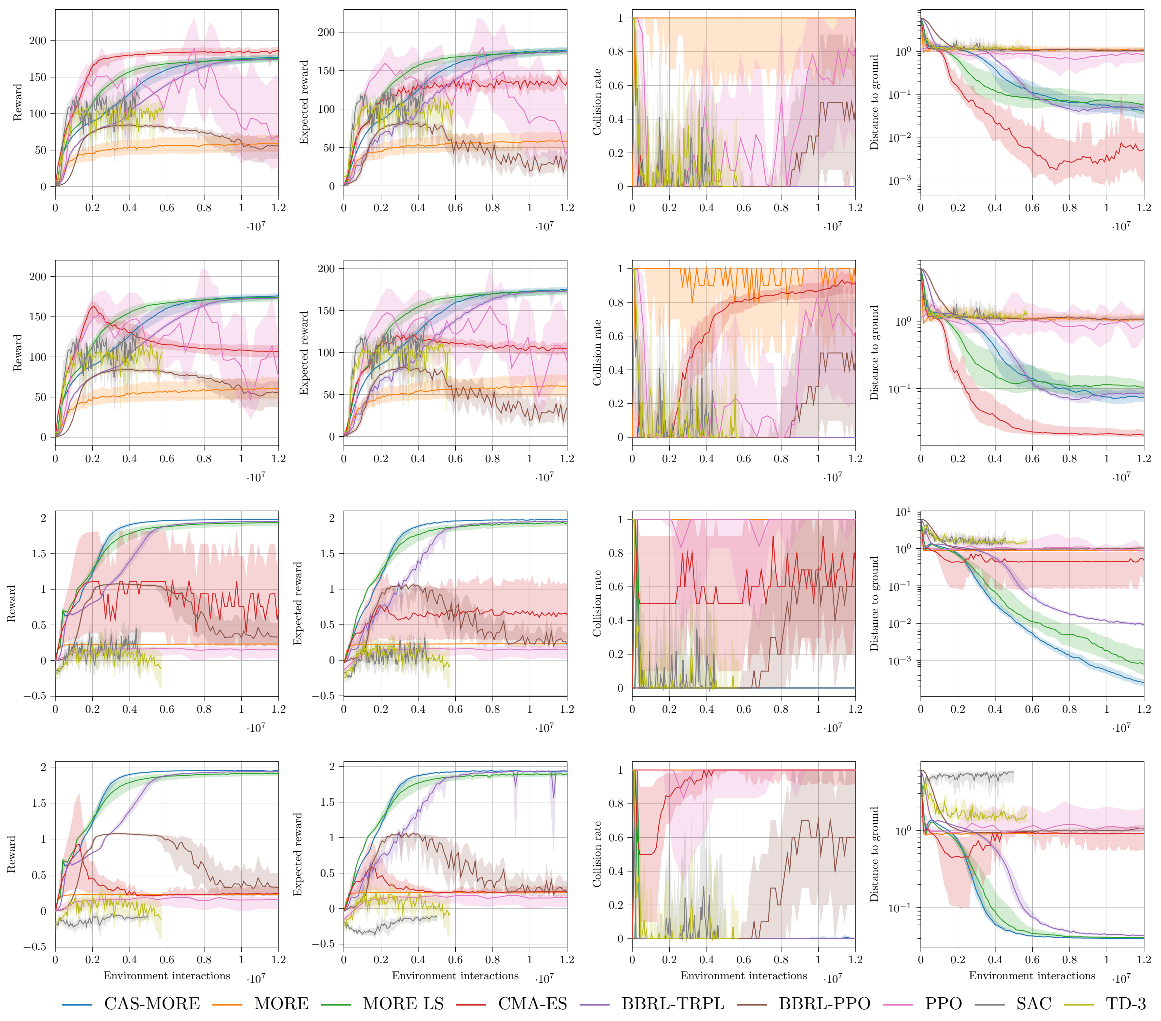


Figure 7: This figure shows the results of the hole-reacher experiment. Plotted is the expected performance of the mean in the left column, the mean of episode returns of the samples drawn in each iteration in the center left column, the percentage of trajectories that led to collisions with the ground in the center right column, and the distance of the end-effector at the end of the trajectory in the right column. The first row contains the results for the noise-less experiment with dense step-based reward, while the second row shows the results of the noisy experiment with dense step-based reward. The third and fourth row show the results for the noiseless and noisy experiments using a sparse episodic reward, respectively.

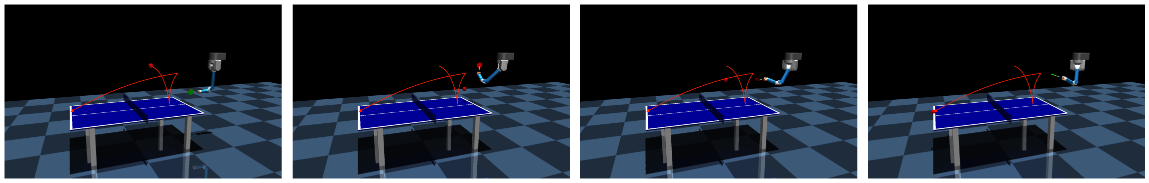


Figure 8: A successful episode of the table tennis task learned by CAS-MORE.

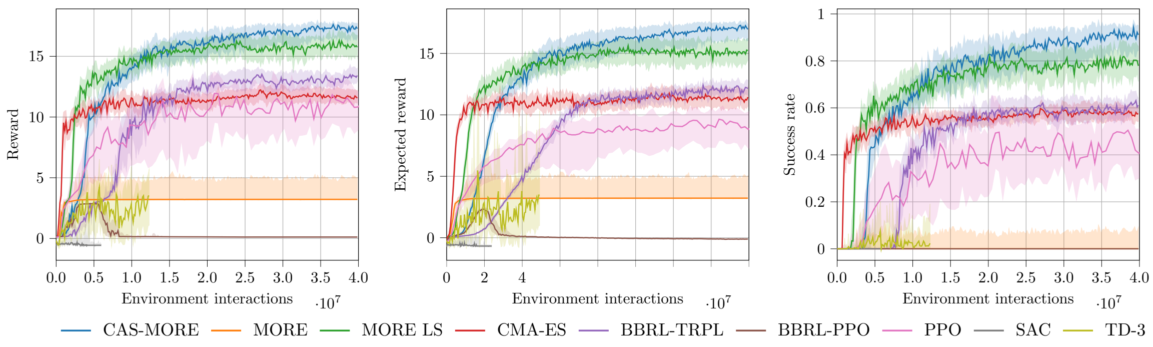


Figure 9: This figure shows the results of the table tennis experiment. Plotted is the expected performance of the mean in the left column, the mean of episode returns of the samples drawn in each iteration in the center column, and the percentage of trajectories that led to ball landing points within the last 10cm of the table in the right column.

initialized as zero, so that the robot needs to learn the movement from scratch. The reward function is composed of three stages. First, the distance between the ball and the racket is minimized to enforce hitting of the ball. Second, a term is added to minimize the distance between the landing point of the ball and the far edge of the table. Once the ball lands on the table, the reward increases with the distance of the ball. Note, that different from the hole-reacher task, the reward for this task is non-Markovian as it depends on the whole trajectory (Otto et al., 2023). In order to provide a step-based based reward for the step-based RL algorithms, we run the simulation until the racket touches the ball while only return the action cost. The rest of the episode is then simulated without agent interaction and presented to the learning algorithm as one last time-step with the task reward as in Otto et al. (2023). For more details on the environment, we refer the reader to Appendix E.2. The reward function differs from Otto et al. (2023) to showcase the risk awareness of the optimization algorithms, as the reward drops as soon as the ball flies past the table.

The results of the experiment are presented in Figure 9. We can again see that CAS-MORE and MORE using the novel model learning approach produce the best results, this time with a distinct advantage over both, step-based RL algorithms PPO, TD3 and SAC, as well as gradient based episodic RL algorithms BBRL TRPL and BBRL PPO. Only CAS-MORE is able to consistently return around 90% of the balls into the target zone towards the edge of the table. CMA-ES and BBRL TRPL only produce distributions that are able to return around 60% of the balls.

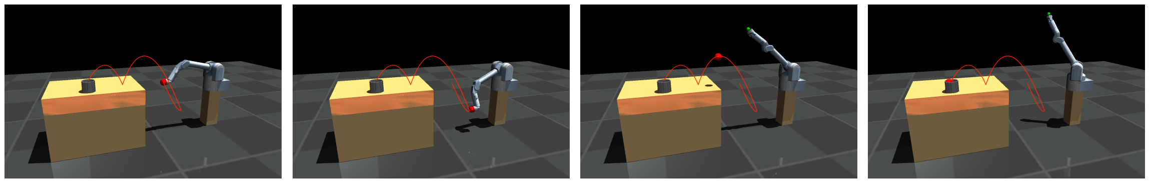


Figure 10: A successful episode of the beerpong task learned by CAS-MORE.

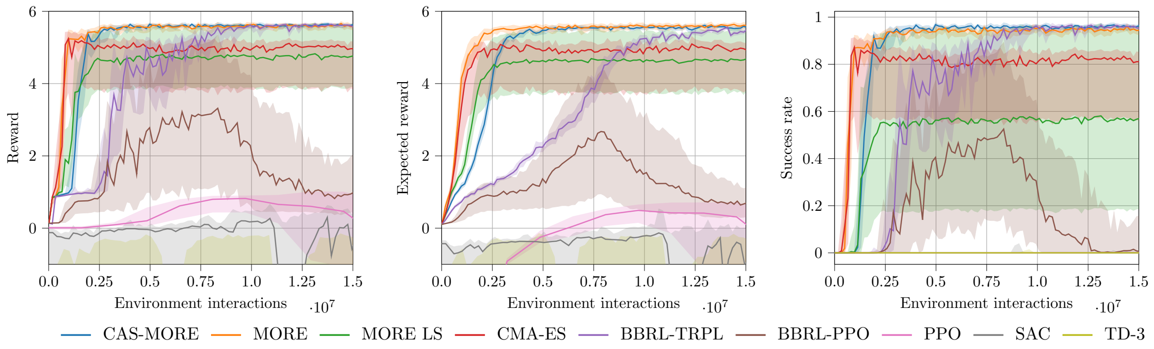


Figure 11: This figure shows the results of the beerpong experiment. Plotted is the expected performance of the mean in the left column, the mean of episode returns of the samples drawn in each iteration in the center column, and the percentage of successful throws where the ball landed in the cup in the right column.

6.2.3 Beerpong

We use the same simulated robot as before to throw a ball towards a table where it first needs to bounce at least once and then fly into a cup. The task is depicted in in Figure 10. As the robot has no actuated end-effector and the ball is directly attached to the robot's last joint, we only need to actuate the first 6 joints, resulting in a 12 dimensional search space for the episodic RL algorithms. In this experiment, the ball release is noisy and designed in a way that it is not possible to find a conservative strategy. The noise will always cause some trials to miss the cup. The results can be found in Figure 11.

Overall, CAS-MORE again provides the best trade-off between convergence speed and task performance. Interestingly, the original MORE performs better than MORE using the least squares model while the step-based reinforcement learning tasks fail to find any good strategy. This experiment also shows the benefits of trust-region optimization, as BBRL-PPO starts to approach a good strategy but then fails due to, presumably, too aggressive policy updates.

6.2.4 Hopper Jump

The last task we consider is a modified version of the Hopper task from OpenAI gym (Brockman et al., 2016) where the agent is rewarded for jumping onto a box. To this end, we place a box in front of the Hopper and use an updated reward function Appendix E.4. In the simulation, the initial distance to the agent and height of the box are subject to noise. For the episodic RL experiments, we choose 3 basis functions for each of the 3 degrees of

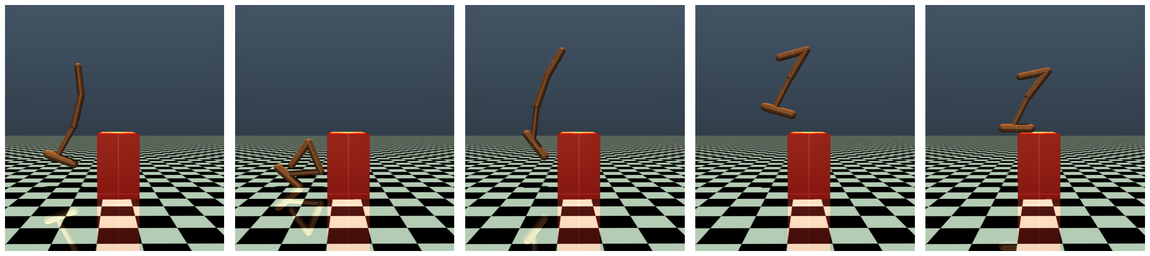


Figure 12: A successful episode of the hopper jump task learned by CAS-MORE. At first, the agent fully bends to then release into a fully extended jump. This way, it can reach a safe height not risking contact with the box's edge.

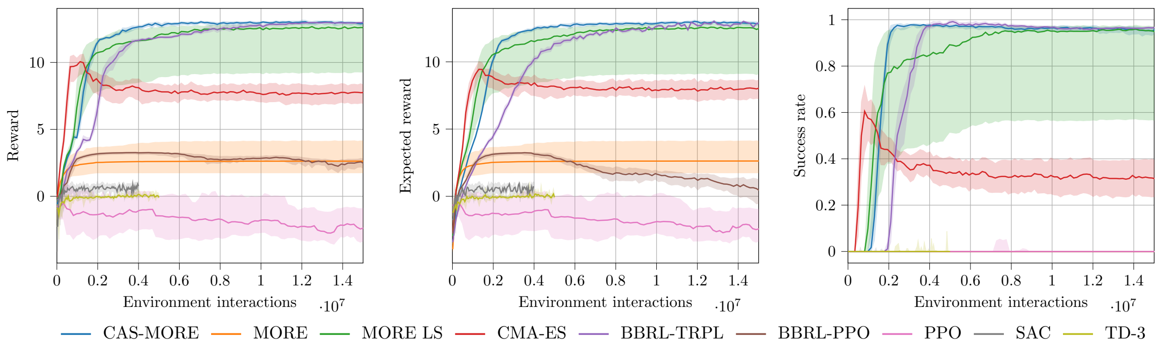


Figure 13: This figure shows the results of the hopper jump experiment. Plotted is the expected performance of the mean in the left column, the mean of episode returns of the samples drawn in each iteration in the center column, and the percentage of successful throws where the ball landed in the cup in the right column.

freedom, resulting in 9 parameters to be optimized. Compared to the original version of the Hopper task, it is very difficult for step-based reinforcement learning algorithms, since a successful strategy requires broad exploration and involves tucking by bending the knee joint and an explosive move upwards to gain enough momentum to reach the top of the box. An illustration of this behavior can be found in Figure 12

The results are presented in Figure 13. We first observe, that only CAS-MORE, MORE LS and BBRL TRPL are able to solve the task. Again, CAS-MORE finds good solutions more robustly than MORE LS, as indicated by the confidence interval, and more efficient in terms of samples and in terms of computation time compared to BBRL TRPL. CMA-ES again quickly optimizes towards a solution but soon fails once the noise has a direct impact on the trajectory of the agent. All other algorithms hardly find policies that produce jumping strategies.

7. Conclusion

In this paper, we presented CAS-MORE, a new version of Model-based Relative Entropy Stochastic Search, based on a coordinate ascent strategy on the mean and covariance of the search distribution and an adaptive entropy schedule. Additionally, we provide an improved model fitting process that is computationally more efficient and show how to deal with objective functions that produce large outliers. We show how the parameter updates follow the direction of the natural gradient and, as the model estimation is not based on rankings of objective function values, we truly optimize the expected fitness under the search distribution. The result is a robust risk-aware optimization which we show to outperform state of the art algorithms such as CMA-ES on stochastic search task, as well as episodic and step-based reinforcement learning tasks, especially on noisy episodic reinforcement learning tasks.

Limitations and Future Work Arguably the biggest limitation of CAS-MORE is the restriction to the non-contextual setting. However, we believe that CAS-MORE still is a valuable tool in the prototyping stage of a more complex contextual problem, or if the problem at hand is non-contextual. Here, the robustness and speed of the algorithm provide a large benefit over more complex algorithms. In addition, the algorithm can easily be extended for small context spaces with low-dimensional contexts using a linear mapping from the context to the mean.

Acknowledgments

The authors acknowledge support by the state of Baden-Württemberg through bwHPC. Research that lead to this work was funded by the Federal Ministry of Education and Research (BMBF) and the state of Hesse as part of the NHR Program. Further, part of this work was performed on the HoreKa supercomputer funded by the Ministry of Science, Research and the Arts Baden-Württemberg and by the Federal Ministry of Education and Research.

Appendix A. Derivation of CA-MORE Dual

Before we solve the optimization problems, we will first state the closed-form solution of the objective and KL-divergence using a quadratic model under multi-variate Gaussian distributions. The solution to the objective is given by

$$\int_{\boldsymbol{x}} \pi(\boldsymbol{x})\hat{f}(\boldsymbol{x})\mathrm{dx} = -\frac{1}{2}\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{\mu} - \frac{1}{2}\mathrm{tr}(\boldsymbol{A}\boldsymbol{\Sigma}) + \boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{a} + a_0$$

and the KL-divergence between two Gaussian distributions is given by

$$\begin{aligned}\mathrm{KL}(\pi \parallel \pi_t) =& \frac{1}{2}\left\{(\boldsymbol{\mu}_t - \boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}_t^{-1}(\boldsymbol{\mu}_t - \boldsymbol{\mu})\right.\\ &\left. + \mathrm{tr}(\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\Sigma}) - k + \log|\boldsymbol{\Sigma}_t| - \log|\boldsymbol{\Sigma}|\right\}.\end{aligned}$$

A.1 Mean Update

Setting $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_t$, the optimization problem is given by

$$\begin{aligned} \underset{\boldsymbol{\mu}}{\text{maximize}} \quad & -\frac{1}{2}\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{a} \\ \text{subject to} \quad & \frac{1}{2}(\boldsymbol{\mu}_t - \boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}_t^{-1}(\boldsymbol{\mu}_t - \boldsymbol{\mu}) \leq \epsilon_\mu \end{aligned}$$

and the Lagrangian is given by

$$\begin{aligned} L(\boldsymbol{\mu}, \lambda) = & -\frac{1}{2}\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{a} \\ & + \lambda\left(\epsilon_\mu - \frac{1}{2}(\boldsymbol{\mu}_t - \boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}_t^{-1}(\boldsymbol{\mu}_t - \boldsymbol{\mu})\right) \end{aligned}$$

where λ is a Lagrangian multiplier. The optimal solution $\boldsymbol{\mu}^*$ in terms of the Lagrangian multipliers can be found by differentiating L with respect to $\boldsymbol{\mu}$ and setting it to 0, i.e,

$$\frac{\partial L}{\partial \boldsymbol{\mu}} = -\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{a} + \lambda\boldsymbol{\Sigma}_t^{-1}(\boldsymbol{\mu}_t - \boldsymbol{\mu}) \overset{!}{=} 0.$$

Using the solution λ^*,

$$\boldsymbol{\mu}^* = \underbrace{(\lambda^*\boldsymbol{\Sigma}_t^{-1} + \boldsymbol{A})}_{\boldsymbol{M}_\mu(\lambda^*)}{}^{-1}\underbrace{(\lambda^*\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\mu}_t + \boldsymbol{a})}_{\boldsymbol{m}_\mu(\lambda^*)} = \boldsymbol{M}_\mu(\lambda^*)^{-1}\boldsymbol{m}_\mu(\lambda^*)$$

After rearranging terms, the dual problem for the mean is given by

$$\begin{aligned} g_\mu(\lambda) = \lambda\epsilon_\mu + \frac{1}{2}\Big(&\boldsymbol{m}_\mu(\lambda)^{\mathrm{T}}\boldsymbol{M}_\mu(\lambda)^{-1}\boldsymbol{m}_\mu(\lambda) \\ & - \lambda\boldsymbol{m}_t^{\mathrm{T}}\boldsymbol{M}_t^{-1}\boldsymbol{m}_t\Big). \end{aligned}$$

A.2 Covariance Update

Analogously, we set $\boldsymbol{\mu} = \boldsymbol{\mu}_t$. The optimization problem for the covariance is given by

$$\begin{aligned} \underset{\boldsymbol{\Sigma}}{\text{maximize}} \quad & -\frac{1}{2}\mathrm{tr}(\boldsymbol{A}\boldsymbol{\Sigma}) \\ \text{subject to} \quad & \frac{1}{2}\left(\mathrm{tr}(\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\Sigma}) - k + \log|\boldsymbol{\Sigma}_t| - \log|\boldsymbol{\Sigma}|\right) < \epsilon_\Sigma \end{aligned}$$

and the Lagrangian for the covariance matrix optimization is given by

$$\begin{aligned} L(\boldsymbol{\Sigma}, \nu) = & -\frac{1}{2}\mathrm{tr}(\boldsymbol{A}\boldsymbol{\Sigma}) \\ & + \nu\left(\epsilon_\Sigma - \frac{1}{2}\left(\mathrm{tr}(\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\Sigma}) - k + \log|\boldsymbol{\Sigma}_t| - \log|\boldsymbol{\Sigma}|\right)\right) \end{aligned}$$

where ν is again a Lagrangian multiplier. The optimal solution $\boldsymbol{\Sigma}^*$ can be found analogously and is given by

$$\frac{\partial L_\Sigma}{\partial \boldsymbol{\Sigma}} = -\frac{1}{2}\boldsymbol{A} - \frac{1}{2}\nu\boldsymbol{\Sigma}_t^{-1} + \frac{1}{2}\nu\boldsymbol{\Sigma}^{-1} \overset{!}{=} 0.$$

With the solution $\nu^a st$,

$$\mathbf{\Sigma}^* = \underbrace{\left((\nu^* \mathbf{\Sigma}_t^{-1} + \boldsymbol{A})/\nu^*\right)}_{\boldsymbol{S}(\nu^*)}{}^{-1} = \boldsymbol{S}(\nu^*)^{-1}.$$

After rearranging terms again, the dual for the covariance is given by

$$\mathbf{\Lambda}(\nu) = \frac{\nu \mathbf{\Sigma}_t^{-1} + \boldsymbol{A}}{\nu}.$$

Appendix B. Robust Target Normalization

In reinforcement learning, a reward function that accurately describes the task and is easy to optimize is not always given. Here, we want to demonstrate that even with a reward function that includes large jumps, using a robust target normalization scheme leads to good results. To this end, we use a different reward function formulation for the hole-reacher task from Section 6.2.1. The reward in this case is defined as the negative squared distance to a target point at the bottom of the hole minus a large penalty whenever the robot hits the ground. We leave the depth at -1 to focus on the effects of target normalization. Figure 14 shows the negative reward (the cost) on a log-scale and we compare CAS-MORE using mean/std normalization and robust mean/std normalization. A cost of 1 corresponds to the end-effector of the robot moving close to the entrance of the hole but failing to reach inside. Only robust normalization is able to capture both, the task reward and the penalty, and guides the robot to reach down the hole. We provide pseudo-code for the robust target normalization technique in Algorithm 1.

Algorithm 1 Robust target normalization

```
1:  procedure NORMALIZE($\mathcal{Y}$)                                   ▷ Robust normalization of targets
2:      $\bar{y}_{\mathcal{Y}} \leftarrow \frac{1}{|\mathcal{Y}|}\sum_q^{|\mathcal{Y}|} y_q$
3:      $\sigma_{\mathcal{Y}} \leftarrow \sqrt{\frac{1}{|\mathcal{Y}|}\sum_q^{|\mathcal{Y}|}(y_q - \bar{y}_{\mathcal{Y}})^2}$
4:      $y \leftarrow \frac{y - \bar{y}_{\mathcal{Y}}}{\sigma_{\mathcal{Y}}}$                          ▷ Standardize all elements in $\mathcal{Y}$
5:      $\mathcal{I} \leftarrow -v_{\text{clip}} < y < v_{\text{clip}}$ ▷ Boolean mask of all elements in $\mathcal{Y}$ that are in $(-v_{\text{clip}}, v_{\text{clip}})$
6:      $\mathcal{S} \leftarrow \{y \in \mathcal{Y} \mid -v_{\text{clip}} < y < v_{\text{clip}}\}$       ▷ All elements in $\mathcal{Y}$ that are in $(-v_{\text{clip}}, v_{\text{clip}})$
7:      $k \leftarrow \text{Kurt}[\mathcal{S}] - 3$                      ▷ Excess kurtosis of the elements in $\mathcal{S}$
8:      if $k > 0.55$ and $\sigma_{\mathcal{Y}} \neq 1$ then
9:          $\mathcal{Y}(\mathcal{I}) \leftarrow$ NORMALIZE($\mathcal{S}$)                    ▷ Normalize elements in $\mathcal{S}$
10:     end if
11:     $\mathcal{Y} \leftarrow \text{clip}(\mathcal{Y}, \min(\mathcal{Y}(\mathcal{I})), \max(\mathcal{Y}(\mathcal{I})))$
12:     return $\mathcal{Y}$
13: end procedure
```

Appendix C. Hyper-Parameters

We provide default hyper-parameters for CAS-MORE in terms of the problem dimensionality n in Table 2 which we empirically found to work well over all benchmark functions. For

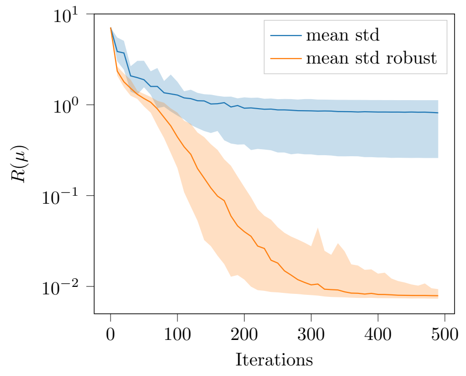


Figure 14: Comparison of mean/std normalization and robust normalization on a penalty based reward function for the hole-reaching task.

Table 2: Empirically found default hyper-parameters for CAS-MORE based on the problem dimensionality n.

Parameter	Default Value
K: Population size	$4 + \lfloor 3 \log(n) \rfloor$
$Q_{\max}$: Maximum queue size	$\max\{\lceil 1.5(1 + n + n(n+1)/2) \rceil,\ 8(n+1)\}$
ϵ_μ: Trust-region for the mean	0.5
ϵ_Σ: Trust-region for the covariance	$\frac{1.5}{10+n^{1.5}}$
c_σ: Smoothing factor of evolution path	$\frac{1}{2+n^{0.75}}$
v_{clip}: Clip value for robust normalization	3
Excess kurtosis threshold	0.55

some functions, a higher bound on the mean often leads to quicker convergence but may result in divergence for others.

Table 3 provides the chosen hyper-parameters for the step-based deep RL experiments (PPO, SAC, TD3), as well as the BBRL experiments. Note, that samples per iteration corresponds to tuples (s, a, r, s') in the step-based case, and whole trajectories in the BBRL case.

Appendix D. Black-box Optimization Benchmarks

Results from experiments according to Hansen et al. (2016b) and Hansen et al. (2016a) on the benchmark functions given in Finck et al. (2009); Hansen et al. (2009a) are presented in Figures 15 to 17. The experiments were performed with COCO (Hansen et al., 2020), version 2.4.1.1, the plots were produced with version 2.4.1.1. The **expected runtime (ERT)**, used in the figures and tables, depends on a given target function value, $f_t = f_{\text{opt}} + \Delta f$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach f_t, summed over all trials and

Table 3: Hyper-parameters for the deep RL and BBRL experiments.

	PPO	SAC	TD3	BBRL-PPO	BBRL-TRPL
samples per iteration	16000	1	1	64	64
GAE λ	0.95	n.a.	n.a.	n.a.	n.a.
discount factor	0.99	0.99	0.99	n.a.	n.a.
ϵ_μ	n.a.	n.a.	n.a.	n.a.	0.05
ϵ_Σ	n.a.	n.a.	n.a.	n.a.	0.005
optimizer	adam	adam	adam	adam	adam
epochs	10	n.a.	n.a.	100	100
learning rate	3e-4	3e-4	3e-4	1e-3	3e-4
use critic	True	True	True	False	False
epochs critic	10	n.a.	n.a.	n.a.	n.a.
learning rate critic	3e-4	3e-4	3e-4	n.a.	n.a.
learning rate alpha	n.a.	3e-4	n.a.	n.a.	n.a.
warm up steps	0	10000	25000	0	0
minibatch size	512	n.a.	n.a.	64	64
batch size	n.a.	256	256	n.a.	n.a.
buffer size	n.a.	1e6	1e6	n.a.	n.a.
polyak_weight	n.a.	5e-3	5e-3	n.a.	n.a.
normalized observations	True	False	False	False	False
normalized rewards	True	False	False	False	False
critic clip	0.2	n.a.	n.a.	0.2	n.a.
importance ratio clip	0.2	n.a.	n.a.	0.2	n.a.
hidden layers	[256, 256]	[256, 256]	[256, 256]	n.a	n.a
hidden layers critic	[256, 256]	[256, 256]	[256, 256]	n.a.	n.a.
hidden activation	tanh	ReLU	ReLU	n.a.	n.a.
initial std	0.6	1.0	0.1	1.0	1.0

divided by the number of trials that actually reached f_t (Hansen et al., 2012; Price, 1997). **Statistical significance** is tested with the rank-sum test for a given target Δf_t using, for each trial, either the number of needed function evaluations to reach Δf_t (inverted and multiplied by -1), or, if the target was not reached, the best Δf-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration.

Appendix E. Episodic RL

In this section, we provide additional information and reward functions for the episodic RL tasks. The action cost for all tasks is given by

$$\tau_t = \sum_i^K (a_t^i)^2.$$

E.1 Holereaching

The reward function is composed of two phases. In the first phase, the distance $d_g = \|\boldsymbol{p}_t - \boldsymbol{p}_g\|$ of the position of the end-effector $\boldsymbol{p}_t = (x_{\text{ee},t}, y_{\text{ee},t})$ to a goal point $\boldsymbol{p}_g = (2, -0.1)$ below the entrance of the hole is minimized. Afterwards, the reward scales linearly with the absolute value $y_{\text{ee},t}$ of the end-effector. The task ends if either $t = 200$ or a collision happens. In the deterministic case, the hole has a depth of $1\,\text{m}$, while in the stochastic case, the depth is sampled uniformly from $\mathcal{U}(0.98\,\text{m}, 1.02\,\text{m})$. The task reward is given by

$$R_{\text{task}} = \begin{cases} c_{\text{coll}} \exp(-d_g) & \text{if a collision happened or } t = T, \\ \exp(-d_g) & \text{if } y_{\text{ee},t} \geq 0 \text{ and no collision happened,} \\ 1 + |y_{\text{ee},t}| & \text{if } y_{\text{ee},t} < 0 \text{ and no collision happened} \end{cases}$$

where $c_{\text{coll}} = 0.25$ if there is a collision and $c_{\text{coll}} = 1$ otherwise.

Dense reward. The dense reward is given in each time-step by

$$r_t = R_{\text{task}} - 0.01\tau_t$$

Sparse reward. The sparse reward only returns the task reward in the terminal time-step T which can either be the last time-step of the episode or the time-step where a collision happens and is given by

$$r_t = \begin{cases} -0.001\tau_t & \text{if } t < T, \\ R_{\text{task}} - 0.001\tau_t & \text{if } t = T. \end{cases}$$

E.2 Table Tennis

In the table tennis task, the episode starts with a fixed ball position and velocity. In order to add stochasticity to the environment, we add noise to the initial velocity in x-direction (i.e., along the long edge of the table tennis table). The task is to hit the ball so that it lands close to the opponent's side short edge of the table. The reward function is non Markovian and is based on the minimum distance $d_{b,r} = \min_t \|\boldsymbol{p}_{b,t} - \boldsymbol{p}_{r,t}\|$ between the ball position $\boldsymbol{p}_{b,t}$ and the racket position $\boldsymbol{p}_{r,t}$, the minimum distance $d_{b,g} = \min_t \|\boldsymbol{p}_{b,t} - \boldsymbol{p}_g\|$ between the ball position and a goal point $\boldsymbol{p}_g = (-1.3, 0, 0.77)$, and the distance $d_{l,e} = |p_{l,x} - p_{g,x}|$ between the x-coordinate of the ball landing position $\boldsymbol{p}_l$ and x-coordinate of the opponent's edge $p_{g,x} = -1.3$ within an episode. We use the transformation

$$\rho(x) = 1 - \frac{x}{1 + x}$$

to convert large distances to a value of 0 and small distances to a value of 1. The task reward is given as

$$R_{\text{task}} = \begin{cases} 0.2\rho(d_{b,r}) & \text{if ball was not hit,} \\ 0.5\rho(d_{b,r}) + \rho(d_{b,g}) & \text{if ball was hit but landed on floor,} \\ 2\rho(d_{b,r}) + 3\rho(d_{l,e}) & \text{if ball was hit, landed on the table and } p_{l,x} \geq -1.25 \\ 5\rho(d_{b,r}) + 10|p_{l,x}| & \text{if ball was hit, landed on the table and } p_{l,x} < -1.25 \end{cases},$$

and we consider a trajectory to be successful if the ball lands within 10cm of the opponent's side edge of the table. We use the sparse-in-time reward

$$r_t = \begin{cases} -0.001\tau_t & \text{if } t < T, \\ R_{\text{task}} - 0.001\tau_t & \text{if } t = T. \end{cases}$$

E.3 Beerpong

In the table beerpong task, the episode starts with the ball attached to the end-effector of the arm. In order to add stochasticity to the environment, we add noise to the velocity at a pre-defined fixed time-step of the release of the ball. For a successful throw, the ball first needs to bounce once on the table and then land inside the cup. The reward function is again non Markovian and is based on the final distance $d_{b,c,T} = \|\boldsymbol{p}_{b,T} - \boldsymbol{p}_c\|$, and on the minimum distance $d_{b,c} = \min_t \|\boldsymbol{p}_{b,t} - \boldsymbol{p}_c\|$ between the ball position $\boldsymbol{p}_{b,t}$ and the center of the cup $\boldsymbol{p}_c$ within an episode. With the same transform $\rho(x)$ as before, the task reward is given by

$$R_{\text{task}} = \begin{cases} 0.2\rho(d_{b,c}) + 0.1\rho(d_{b,c,T}) & \text{if ball first has contact with anything but the table,} \\ \rho(d_{b,c}) + 0.5\rho(d_{b,c,T}) & \text{if ball first has contact w. the table but is not in cup,} \\ \rho(d_{b,c}) + 2\rho(d_{b,c,T}) + 1 & \text{if ball is in cup, without contact with the table,} \\ \rho(d_{b,c}) + 2\rho(d_{b,c,T}) + 3 & \text{if ball is in cup and had contact with the table before.} \end{cases}$$

The sparse-in-time reward is given by

$$r_t = \begin{cases} -0.1\tau_t & \text{if } t < T, \\ R_{\text{task}} - 0.1\tau_t & \text{if } t = T. \end{cases}$$

E.4 Hopper Jump

The task reward for the hopper task is composed of a reward term for jumping as high as possible and a distance minimization term to the center of top of the box. Additionally, bonuses are added for successfully landing upright and landing on the box. High velocities, as well as joint angles that lead to falling over on the box result in a penalty. We record the maximum height $h_{\max}$ of the agent's torso during an episode, the minimum distance $d_{f,c} = \min_t \|\boldsymbol{p}_{f,t} - \boldsymbol{p}_c\|$ and final distance $d_{f,c,T} = \|\boldsymbol{p}_{f,T} - \boldsymbol{p}_c\|$ between the agent's foot position $\boldsymbol{p}_f$ and the top center of the box $\boldsymbol{p}_c$. The individual reward terms are given as

$$\begin{aligned}
R_{\text{height}} &= \begin{cases} 5h_{\max} & \text{if } h_{\max} < 2, \\ 10 + h_T & \text{if } h_{\max} \geq 2 \text{ and agent did not fall after landing on the box,} \\ 2h_{\max} & \text{if agent is on box but falling over,} \end{cases} \\
R_{\text{dist}} &= \begin{cases} -5 & \text{if } h_{\max} < 2, \\ -5d_{f,c,T} & \text{if } h_{\max} \geq 2, \end{cases} \\
R_{\text{min dist}} &= -2d_{f,c} \\
R_{\text{healthy}} &= \begin{cases} 1 & \text{if } z_T \in [0.7, \infty],\ \phi \in [-\infty, \infty] \text{ and } \theta, \dot{\theta}, \in [-100, 100], \\ 0 & \text{else,} \end{cases} \\
R_{\text{box}} &= \begin{cases} 5 & \text{if agent is on box and } h_T > 1.6, \\ 0 & \text{else,} \end{cases} \\
R_{\text{box vel}} &= \begin{cases} -\min(10v_x^2, 1) & \text{if agent is on box,} \\ 0 & \text{else.} \end{cases}
\end{aligned}$$

The task reward is given as

$$R_{\text{task}} = R_{\text{height}} + R_{\text{dist}} + R_{\text{min dist}} + R_{\text{healthy}} + R_{\text{box}} + R_{\text{box vel}}.$$

and the total reward for each time step is given as

$$r_t = \begin{cases} -10^{-4}\tau_t & \text{if } t < T, \\ R_{\text{task}} - 10^{-4}\tau_t & \text{if } t = T. \end{cases}$$

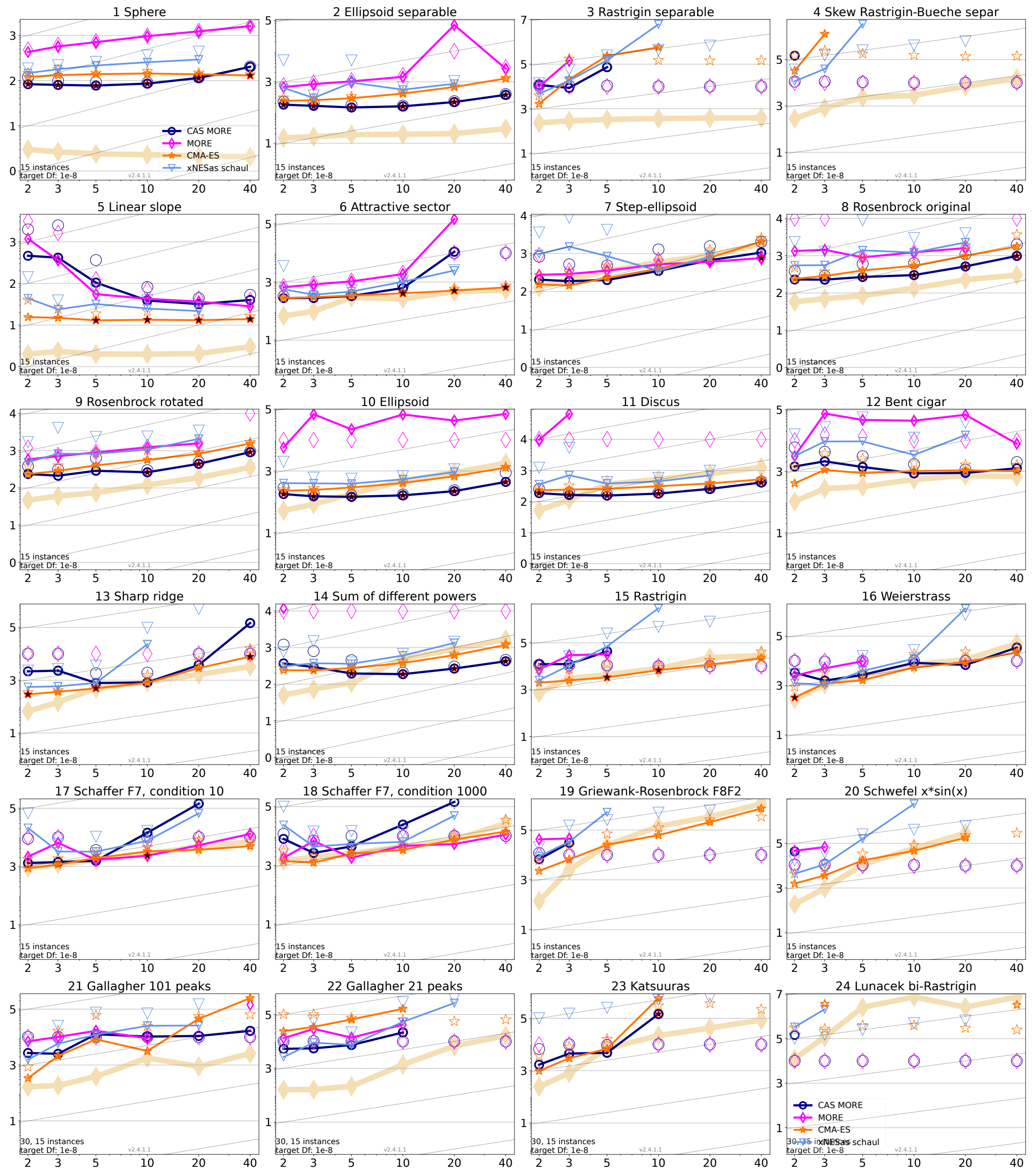


Figure 15: Expected running time (ERT in number of f-evaluations as $\log_{10}$ value), divided by dimension for target function value 10^{-8} versus dimension. Slanted grid lines indicate quadratic scaling with the dimension. Different symbols correspond to different algorithms given in the legend of f_1 and f_{24}. Light symbols give the maximum number of function evaluations from the longest trial divided by dimension. Black stars indicate a statistically better result compared to all other algorithms with $p < 0.01$ and Bonferroni correction number of dimensions (six). Legend: ○: CAS-MORE, ♢: MORE, ⋆: CMA-ES, ▽: xNESas.

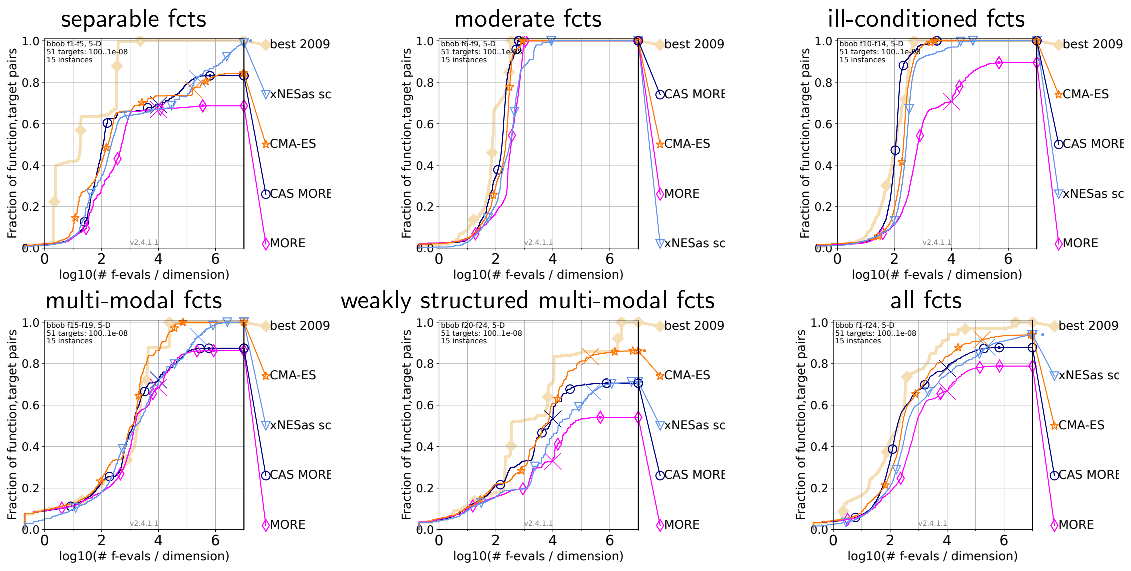


Figure 16: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 5-D. As reference algorithm, the best algorithm from BBOB 2009 is shown as light thick line with diamond markers.

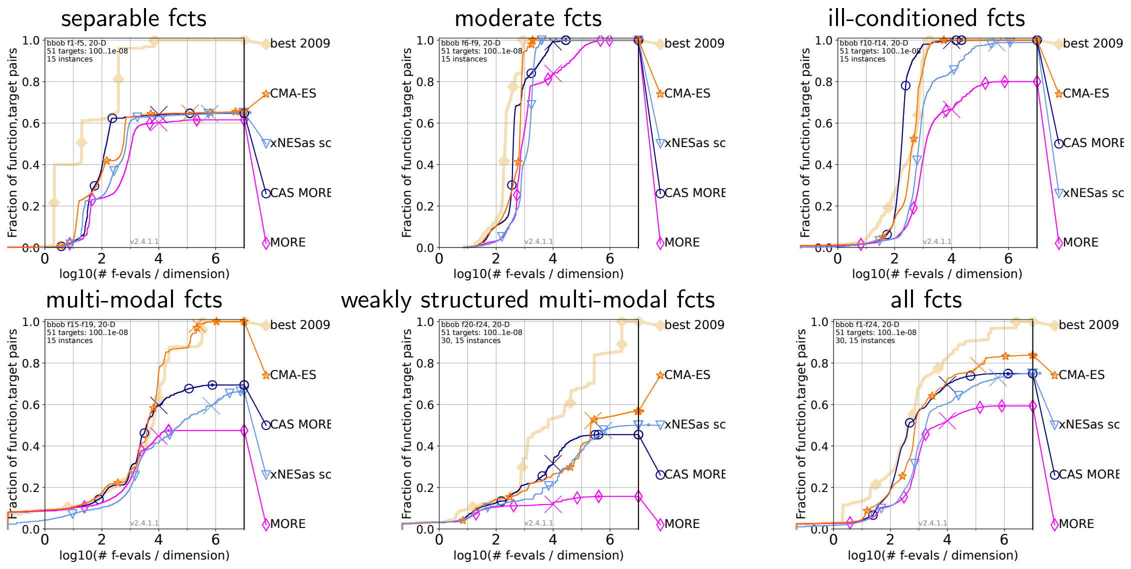


Figure 17: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 20-D. As reference algorithm, the best algorithm from BBOB 2009 is shown as light thick line with diamond markers.

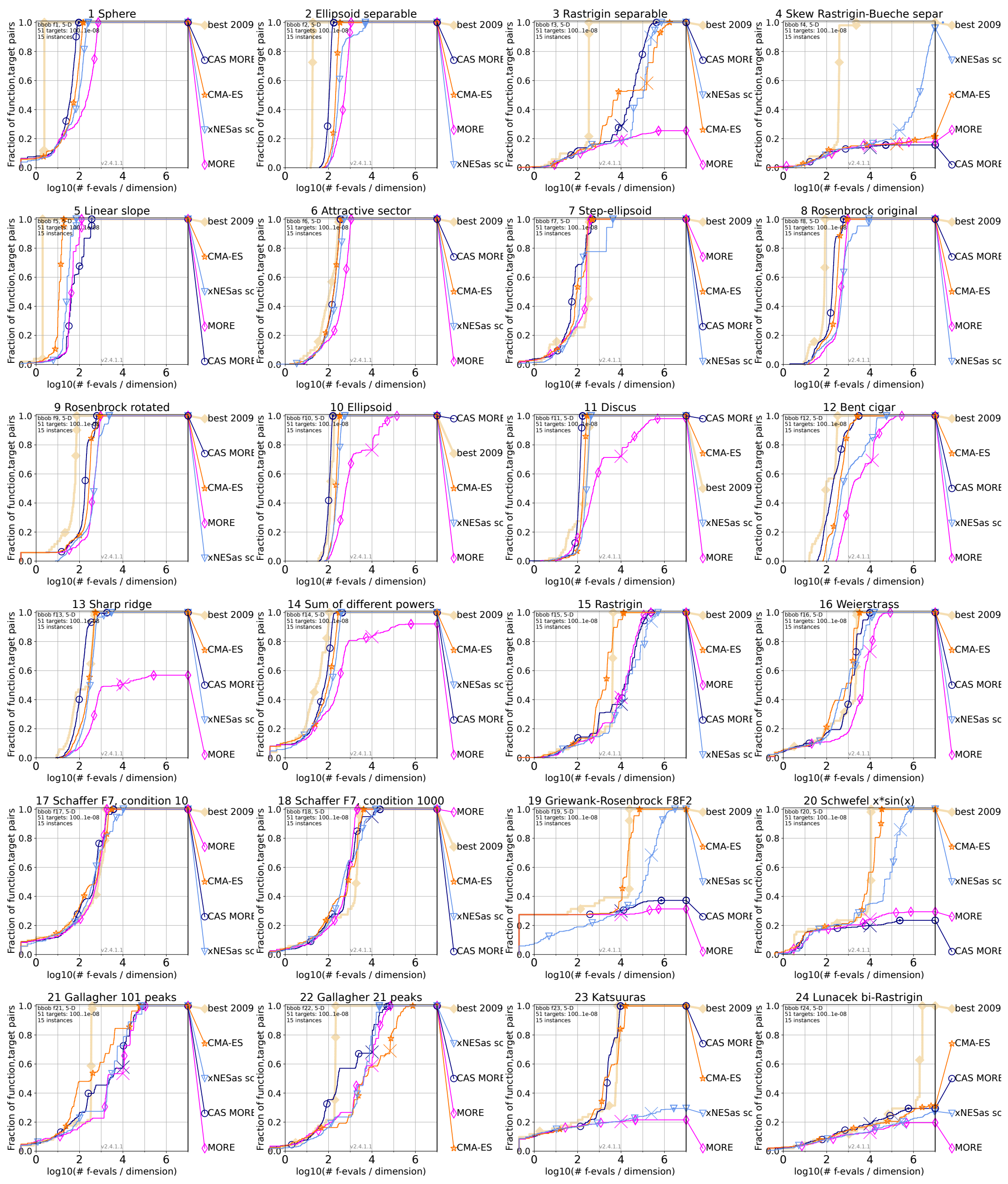


Figure 18: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of objective function evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8..2]}$ in dimension 5.

References

Abbas Abdolmaleki, Rudolf Lioutikov, Jan R Peters, Nuno Lau, Luis Pualo Reis, and Gerhard Neumann. Model-based relative entropy stochastic search. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Abbas Abdolmaleki, Jost Tobias Springenberg, Jonas Degrave, Steven Bohez, Yuval Tassa, Dan Belov, Nicolas Heess, and Martin Riedmiller. Relative entropy regularized policy iteration. *arXiv preprint arXiv:1812.02256*, 2018a.

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018b.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Youhei Akimoto, Yuichi Nagata, Isao Ono, and Shigenobu Kobayashi. Bidirectional relation between cma evolution strategies and natural evolution strategies. In *International Conference on Parallel Problem Solving from Nature*, pages 154–163. Springer, 2010.

Riad Akrour, Abbas Abdolmaleki, Hany Abdulsamad, Jan Peters, and Gerhard Neumann. Model-free trajectory-based policy optimization with monotonic improvement. *The Journal of Machine Learning Research*, 19(1):565–589, 2018.

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

Brandon Amos and Denis Yarats. The differentiable cross-entropy method. In *International Conference on Machine Learning*, pages 291–302. PMLR, 2020.

Oleg Arenz, Gerhard Neumann, and Mingjun Zhong. Efficient gradient-free variational inference using policy search. In *International conference on machine learning*, pages 234–243. PMLR, 2018.

Anne Auger and Nikolaus Hansen. A restart cma evolution strategy with increasing population size. In *2005 IEEE congress on evolutionary computation*, volume 2, pages 1769–1776. IEEE, 2005.

Anne Auger, Marc Schoenauer, and Nicolas Vanhaecke. Ls-cma-es: A second-order algorithm for covariance matrix adaptation. In *International Conference on Parallel Problem Solving from Nature*, pages 182–191. Springer, 2004.

Philipp Becker, Oleg Arenz, and Gerhard Neumann. Expected information maximization: Using the i-projection for mixture density estimation. In *International Conference on Learning Representations*, 2019.

Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies–a comprehensive introduction. *Natural computing*, 1(1):3–52, 2002.

Zdravko I Botev, Dirk P Kroese, Reuven Y Rubinstein, and Pierre L'Ecuyer. The cross-entropy method for optimization. In *Handbook of statistics*, volume 31, pages 35–59. Elsevier, 2013.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Konstantinos Chatzilygeroudis, Roberto Rama, Rituraj Kaushik, Dorian Goepp, Vassilis Vassiliades, and Jean-Baptiste Mouret. Black-box data-efficient policy search for robotics. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 51–58. IEEE, 2017.

Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and trends in Robotics*, 2(1-2):388–403, 2013.

Felix End, Riad Akrour, Jan Peters, and Gerhard Neumann. Layered direct policy search for learning hierarchical skills. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6442–6448. IEEE, 2017.

S. Finck, N. Hansen, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical Report 2009/20, Research Center PPE, 2009. Updated February 2010.

Tobias Glasmachers, Tom Schaul, Sun Yi, Daan Wierstra, and Jürgen Schmidhuber. Exponential natural evolution strategies. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 393–400, 2010.

N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009a. Updated February 2010.

N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2012: Experimental setup. Technical report, INRIA, 2012.

N. Hansen, A Auger, D. Brockhoff, D. Tušar, and T. Tušar. COCO: Performance assessment. *ArXiv e-prints*, arXiv:1605.03560, 2016a.

N. Hansen, T. Tušar, O. Mersmann, A. Auger, and D. Brockhoff. COCO: The experimental procedure. *ArXiv e-prints*, arXiv:1603.08776, 2016b.

N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff. COCO: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 2020. doi: https://doi.org/10.1080/10556788.2020.1808977.

Nikolaus Hansen. Benchmarking a bi-population cma-es on the bbob-2009 function testbed. In *Proceedings of the 11th annual conference companion on genetic and evolutionary computation conference: late breaking papers*, pages 2389–2396, 2009.

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

Nikolaus Hansen. A global surrogate assisted cma-es. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 664–672, 2019.

Nikolaus Hansen, André SP Niederberger, Lino Guzzella, and Petros Koumoutsakos. A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. *IEEE Transactions on Evolutionary Computation*, 13(1): 180–197, 2008.

Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Research Report RR-6829, INRIA, 2009b. URL `https://hal.inria.fr/inria-00362633`.

Nikolaus Hansen, Tea Tusar, Olaf Mersmann, Anne Auger, and Dimo Brockhoff. Coco: The experimental procedure. *arXiv preprint arXiv:1603.08776*, 2016c.

Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. Coco: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36(1):114–144, 2021.

Verena Heidrich-Meisner and Christian Igel. Hoeffding and bernstein races for selecting policies in evolutionary direct policy search. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 401–408, 2009.

John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

Jemin Hwangbo, Christian Gehring, Hannes Sommer, Roland Siegwart, and Jonas Buchli. Rock$*$—efficient black-box optimization for policy learning. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 535–540. IEEE, 2014.

Oscar Ibáñez, Lucia Ballerini, Oscar Cordón, Sergio Damas, and José Santamaría. An experimental study on the applicability of evolutionary algorithms to craniofacial superimposition in forensic identification. *Information Sciences*, 179(23):3998–4028, 2009.

Steven G Johnson. The nlopt nonlinear-optimization package, 2014.

Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

Jakub Kudela. A critical problem in benchmarking and analysis of evolutionary computation methods. *Nature Machine Intelligence*, pages 1–8, 2022.

A Kupcsik, MP Deisenroth, J Peters, and G Neumann. Data-efficient contextual policy search for robot movement skills. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. Bellevue, 2013.

Jeffrey Larson, Matt Menickelly, and Stefan M Wild. Derivative-free optimization methods. *Acta Numerica*, 28:287–404, 2019.

Zhenhua Li and Qingfu Zhang. What does the evolution path learn in cma-es? In *Parallel Problem Solving from Nature–PPSN XIV: 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings 14*, pages 751–760. Springer, 2016.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Ilya Loshchilov, Marc Schoenauer, and Michele Sebag. Alternative restart strategies for cma-es. In *International Conference on Parallel Problem Solving from Nature*, pages 296–305. Springer, 2012a.

Ilya Loshchilov, Marc Schoenauer, and Michele Sebag. Self-adaptive surrogate-assisted covariance matrix adaptation evolution strategy. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 321–328, 2012b.

John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

Michael A Osborne, Roman Garnett, and Stephen J Roberts. Gaussian processes for global optimization. In *3rd international conference on learning and intelligent optimization (LION3)*, pages 1–15, 2009.

Fabian Otto, Philipp Becker, Ngo Anh Vien, Hanna Carolin Ziesche, and Gerhard Neumann. Differentiable trust region layers for deep reinforcement learning. *arXiv preprint arXiv:2101.09207*, 2021.

Fabian Otto, Onur Celik, Hongyi Zhou, Hanna Ziesche, Vien Anh Ngo, and Gerhard Neumann. Deep black-box reinforcement learning with movement primitives. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1244–1265. PMLR, 14–18 Dec 2023.

Joni Pajarinen, Hong Linh Thai, Riad Akrour, Jan Peters, and Gerhard Neumann. Compatible natural gradient policy search. *Machine Learning*, 108(8):1443–1466, 2019.

Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

Kenneth Price. Differential evolution vs. the functions of the second ICEO. In *Proceedings of the IEEE International Congress on Evolutionary Computation*, pages 153–157, Piscataway, NJ, USA, 1997. IEEE. doi: 10.1109/ICEC.1997.592287.

WL1551847 Price. Global optimization by controlled random search. *Journal of optimization theory and applications*, 40(3):333–348, 1983.

Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer, 2004.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

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

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

James C Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*, volume 65. John Wiley & Sons, 2005.

Yi Sun, Daan Wierstra, Tom Schaul, and Jürgen Schmidhuber. Efficient natural evolution strategies. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 539–546, 2009.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1): 949–980, 2014.

Susanne Winter, Bernhard Brendel, Ioannis Pechlivanis, Kirsten Schmieder, and Christian Igel. Registration of ct and intraoperative 3-d ultrasound images of the spine using evolutionary and gradient-based methods. *IEEE Transactions on Evolutionary Computation*, 12(3):284–296, 2008.

Zelda B Zabinsky. Random search algorithms. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.