Maximum Likelihood-Based Online Adaptation of Hyper-Parameters in CMA-ES

Ilya Loshchilov[1], Marc Schoenauer[2,3], Michèle Sebag[3,2],
and Nikolaus Hansen[2,3]

[1] Laboratory of Intelligent Systems,
École Polytechnique Fédérale de Lausanne, Switzerland
`Ilya.Loshchilov@epfl.ch`
[2] TAO Project-team, INRIA Saclay - Île-de-France
[3] Laboratoire de Recherche en Informatique (UMR CNRS 8623)
Université Paris-Sud, 91128 Orsay Cedex, France
`FirstName.LastName@inria.fr`

Abstract. The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is widely accepted as a robust derivative-free continuous optimization algorithm for non-linear and non-convex optimization problems. CMA-ES is well known to be almost parameterless, meaning that only one hyper-parameter, the population size, is proposed to be tuned by the user. In this paper, we propose a principled approach called self-CMA-ES to achieve the online adaptation of CMA-ES hyper-parameters in order to improve its overall performance. Experimental results show that for larger-than-default population size, the default settings of hyper-parameters of CMA-ES are far from being optimal, and that self-CMA-ES allows for dynamically approaching optimal settings.

1 Introduction

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES [5]) is a continuous optimizer which only exploits the ranking of estimated candidate solutions to approach the optimum of an objective function $f : \mathbb{R}^n \to \mathbb{R}$. CMA-ES is also invariant w.r.t. affine transformations of the decision space, explaining the known robustness of the algorithm. An important practical advantage of CMA-ES is that all hyper-parameters thereof are defined by default with respect to the problem dimension n. Practically, only the population size λ is suggested to be tuned by the user, e.g. when a parallelization of the algorithm is considered or the problem at hand is known to be multi-modal and/or noisy [1,8]. Other hyper-parameters have been provided robust default settings (depending on n and λ), in the sense that their offline tuning allegedly hardly improves the CMA-ES performance for unimodal functions. In the meanwhile, for multi-modal functions it is suggested that the overall performance can be significantly improved by offline tuning of λ and multiple stopping criteria [16,11]. Additionally, it is shown that CMA-ES can be improved by a factor up to 5-10 by the use of surrogate models on unimodal ill-conditioned functions [14]. This suggests that the CMA-ES performance can be improved by better exploiting the information in the evaluated samples $(\boldsymbol{x}, f(\boldsymbol{x}))$.

This paper focuses on the automatic online adjustment of the CMA-ES hyper-parameters. The proposed approach, called self-CMA-ES, relies on a second CMA-ES instance operating on the hyper-parameter space of the first CMA-ES, and aimed at increasing the likelihood of generating the most successful samples $\boldsymbol{x}$ in the current generation. The paper is organized as follows. Section 2 describes the original $(\mu/\mu_w, \lambda)$-CMA-ES. self-CMA-ES is described in Section 3 and its experimental validation is discussed in Section 4 comparatively to related work. Section 5 concludes the paper.

2   Covariance Matrix Adaptation Evolution Strategy

The Covariance Matrix Adaptation Evolution Strategy [6,7,5] is acknowledgedly the most popular and the most efficient Evolution Strategy algorithm.

The original $(\mu/\mu_w, \lambda)$-CMA-ES (Algorithm 1) proceeds as follows. At the t-th iteration, a Gaussian distribution $\mathcal{N}\left(\boldsymbol{m}^t, \sigma^{t2}\boldsymbol{C}^t\right)$ is used to generate λ candidate solution $\boldsymbol{x}_k \in \mathbb{R}^n$, for $k = 1 \ldots \lambda$ (line 5):

$$\boldsymbol{x}_k^t = \mathcal{N}\left(\boldsymbol{m}^t, \sigma^{t2}\boldsymbol{C}^t\right) = \boldsymbol{m}^t + \sigma^t\mathcal{N}\left(\boldsymbol{0}, \boldsymbol{C}^t\right), \tag{1}$$

where the mean $\boldsymbol{m}^t \in \mathbb{R}^n$ of the distribution can be interpreted as the current estimate of the optimum of function f, $\boldsymbol{C}^t \in \mathbb{R}^{n \times n}$ is a (positive definite) co-variance matrix and σ^t is a mutation step-size. These λ solutions are evaluated according to f (line 6). The new mean $\boldsymbol{m}^{t+1}$ of the distribution is computed as a *weighted sum* of the best μ individuals out of the λ ones (line 7). Weights $w_1 \ldots w_\mu$ are used to control the impact of selected individuals, with usually higher weights for top ranked individuals (line 1).

The adaptation of the step-size σ^t, inherited from the Cumulative Step-Size Adaptation Evolution Strategy (CSA-ES [6]), is controlled by the evolution path $\boldsymbol{p}_\sigma^{t+1}$. Successful mutation steps $\frac{\boldsymbol{m}^{t+1}-\boldsymbol{m}^t}{\sigma^t}$ (line 8) are tracked in the sampling space, i.e., in the isotropic coordinate system defined by the eigenvectors of the covariance matrix $\boldsymbol{C}^t$. To update the evolution path $\boldsymbol{p}_\sigma^{t+1}$, i) a decay/relaxation factor c_σ is used to decrease the importance of previous steps; ii) the step-size is increased if the length of the evolution path $\boldsymbol{p}_\sigma^{t+1}$ is longer than the expected length of the evolution path under random selection $\mathbb{E}\|\mathcal{N}(\boldsymbol{0}, \mathbf{I})\|$; iii) otherwise it is decreased (line 13). Expectation of $\|\mathcal{N}(\boldsymbol{0}, \mathbf{I})\|$ is approximated by $\sqrt{n}(1 - \frac{1}{4n} + \frac{1}{21n^2})$. A damping parameter d_σ controls the change of the step-size.

The covariance matrix update consists of two parts (line 12): a *rank-one update* [7] and a *rank-μ update* [5]. The rank-one update computes the evolution path $\boldsymbol{p}_c^{t+1}$ of successful moves of the mean $\frac{\boldsymbol{m}^{t+1}-\boldsymbol{m}^t}{\sigma^t}$ of the mutation distribution in the given coordinate system (line 10), along the same lines as the evolution path $\boldsymbol{p}_\sigma^{t+1}$ of the step-size. To stall the update of $\boldsymbol{p}_c^{t+1}$ when σ increases rapidly, a h_σ trigger is used (line 9).

The rank-μ update computes a covariance matrix $\boldsymbol{C}_\mu$ as a weighted sum of the covariances of successful steps of the best μ individuals (line 11). Covariance matrix

---

Algorithm 1. The $(\mu/\mu_w, \lambda)$-CMA-ES

---

1. **given** $n \in \mathbb{N}_+$, $\lambda = 4 + \lfloor 3\ln n \rfloor$, $\mu = \lfloor \lambda/2 \rfloor$, $w_i = \frac{\ln(\mu+\frac{1}{2})-\ln i}{\sum_{j=1}^{\mu}(\ln(\mu+\frac{1}{2})-\ln j)}$ for $i = 1 \dots \mu$,

   $\mu_w = \frac{1}{\sum_{i=1}^{\mu} w_i^2}$, $c_\sigma = \frac{\mu_w+2}{n+\mu_w+3}$, $d_\sigma = 1 + c_\sigma + 2\max(0, \sqrt{\frac{\mu_w-1}{n+1}} - 1)$, $c_c = \frac{4}{n+4}$,

   $c_1 = \frac{2}{(n+1.3)^2+\mu_w}$, $c_\mu = \frac{2(\mu_w-2+1/\mu_w)}{(n+2)^2+\mu_w}$

2. **initialize** $m^{t=0} \in \mathbb{R}^n, \sigma^{t=0} > 0, p_\sigma^{t=0} = \mathbf{0}, p_c^{t=0} = \mathbf{0}, C^{t=0} = \mathbf{I}, t \leftarrow 0$

3. **repeat**

4.   **for** $k = 1, \dots, \lambda$ **do**

5. $x_k = m^t + \sigma^t \mathcal{N}(\mathbf{0}, C^t)$

6. $f_k = f(x_k)$

7. $m^{t+1} = \sum_{i=1}^{\mu} w_i x_{i:\lambda}$ // the symbol $i : \lambda$ denotes i-th best individual on f

8. $p_\sigma^{t+1} = (1-c_\sigma)p_\sigma^t + \sqrt{c_\sigma(2-c_\sigma)}\sqrt{\mu_w} C^{t-\frac{1}{2}} \frac{m^{t+1}-m^t}{\sigma^t}$

9. $h_\sigma = \mathbb{1}_{\|p_\sigma^{t+1}\| < \sqrt{1-(1-c_\sigma)^{2(t+1)}}(1.4+\frac{2}{n+1})\mathbb{E}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|}$

10. $p_c^{t+1} = (1-c_c)p_c^t + h_\sigma\sqrt{c_c(2-c_c)}\sqrt{\mu_w}\frac{m^{t+1}-m^t}{\sigma^t}$

11. $C_\mu = \sum_{i=1}^{\mu} w_i \frac{x_{i:\lambda}-m^t}{\sigma^t} \times \frac{(x_{i:\lambda}-m^t)^T}{\sigma^t}$

12. $C^{t+1} = (1-c_1-c_\mu)C^t + c_1 \underbrace{p_c^{t+1} p_c^{t+1 T}}_{\text{rank−one update}} + c_\mu \underbrace{C_\mu}_{\text{rank−}\mu\text{ update}}$

13. $\sigma^{t+1} = \sigma^t \exp(\frac{c_\sigma}{d_\sigma}(\frac{\|p_\sigma^{t+1}\|}{\mathbb{E}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1))$

14. $t = t + 1$

15. **until** *stopping criterion is met*

C itself is replaced by a weighted sum of the rank-one (weight c_1 [7]) and rank-μ (weight c_μ [5]) updates, with c_1 and c_μ positive and $c_1 + c_\mu \leq 1$.

While the optimal parameterization of CMA-ES remains an open problem, the default parameterization is found quite robust on noiseless unimodal functions [5], which explains the popularity of CMA-ES.

3   The Self-CMA-ES

The proposed self-CMA-ES approach is based on the intuition that the optimal hyper-parameters of CMA-ES at time t should favor the generation of the best individuals at time t, under the (strong) assumption that an optimal parameterization and performance of CMA-ES in each time t will lead to the overall optimal performance.

Formally, this intuition leads to the following procedure. Let θ_f^t denote the hyper-parameter vector used for the optimization of objective f at time t (CMA-ES stores its state variables and internal parameters of iteration t in θ^t and the '.'-notation is used to access them). At time $t+1$, the best individuals generated according to θ_f^t are known to be the top-ranked individuals $x_{1:\lambda}^t \dots x_{\mu:\lambda}^t$, where $x_{i:\lambda}^t$ stands for the i-th best individual w.r.t. f. Hyper-parameter vector θ_f^t would thus have been all the better, if it had maximized the probability of generating these top individuals.

Along this line, the optimization of θ_f^t is conducted using a second CMA-ES algorithm, referred to as *auxiliary* CMA-ES as opposed to the one concerned with the optimization of f, referred to as *primary* CMA-ES. The objective of the auxiliary CMA-ES is specified as follows:

Given: hyper-parameter vector θ_f^i and points $(\boldsymbol{x}_{1:\lambda}^i, f(\boldsymbol{x}_{1:\lambda}^i))$ evaluated by primary CMA-ES at steps $i = 1, \ldots, t$ (noted as $\theta_f^{i+1}.f(\boldsymbol{x}_{1:\lambda})$ in Algorithm 2),

Find: $\theta_f^{t,*}$ such that i) backtracking the primary CMA-ES to its state at time $t-1$; ii) replacing θ_f^t by $\theta_f^{t,*}$, would maximize the likelihood of $\boldsymbol{x}_{i:\lambda}^t$ for $i = 1 \ldots \mu$.

The auxiliary CMA-ES might thus tackle the maximization of $g_t(\theta)$ computed as the weighed log-likelihood of the top-ranked μ_{sel} individuals at time t:

$$g_t(\theta) = \sum_{i=1}^{\mu_{sel}} w_{sel,i} \log \left(P(\boldsymbol{x}_{i:\lambda}^t | \theta_f^t = \theta) \right), \qquad (2)$$

where $w_{sel,i} \geq 0$, $i = 1 \ldots \mu_{sel}$, $\sum_{i=1}^{\mu_{sel}} w_{sel,i} = 1$, and by construction

$$P(\boldsymbol{x}_i | \boldsymbol{m}^t, \boldsymbol{C}^t) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{C}^t|}} \exp \left(-0.5(\boldsymbol{m}^t - \boldsymbol{x}_i^t) \boldsymbol{C}^{t-1} (\boldsymbol{m}^t - \boldsymbol{x}_i^t) \right), \qquad (3)$$

where $\boldsymbol{C}^t$ is the covariance matrix multiplied by σ^{t2} and $|\boldsymbol{C}^t|$ is its determinant.

While the objective function for the auxiliary CMA-ES defined by Eq. (2) is mathematically sound, it yields a difficult optimization problem; firstly the probabilities are scale-sensitive; secondly and overall, in a worst case scenario, a single good but unlikely solution may lead the optimization of θ_f^t astray.

Therefore, another optimization objective $h_t(\theta)$ is defined for the auxiliary CMA-ES, where $h_t(\theta)$ measures the agreement on $\boldsymbol{x}_{i:\lambda}^t$ for $i = 1 \ldots \mu$ between i) the order defined from f; ii) the order defined from their likelihood conditioned by $\theta_f^t = \theta$ (Algorithm 3). Procedure *ReproduceGenerationCMA* in Algorithm 3 updates the strategy parameters described from line 7 to line 14 in Algorithm 1 using already evaluated solutions stored in $\theta_f^t.\boldsymbol{x}_{i:\lambda}$. Line 4 computes the Mahalanobis distance, division by step-size is not needed since only ranking will be considered in line 5 (decreasing order of Mahalanobis distances corresponds to increasing order of log-likelihoods). Line 6 computes a weighted sum of ranks of likelihoods of best individuals.

Finally, the overall scheme of self-CMA-ES (Algorithm 2) involves two interdependent CMA-ES optimization algorithms, where the primary CMA-ES is concerned with optimizing objective f, and the auxiliary CMA-ES is concerned with optimizing objective h_t, that is, optimizing the hyper-parameters of the primary CMA-ES[1]. Note that self-CMA-ES is not *per se* a "more parameterless" algorithm than CMA-ES, in the sense that the user is still invited to modify the population size λ. The main purpose of self-CMA-ES is to achieve the online adaptation of the other CMA-ES hyper-parameters.

---

[1] This scheme is actually inspired from the one proposed for surrogate-assisted optimization [13], where the auxiliary CMA-ES was in charge of optimizing the surrogate learning hyper-parameters.

Algorithm 2. The self-CMA-ES

---
1. $t \leftarrow 1$
2. $\theta_f^t \leftarrow$ InitializationCMA() { primary CMA-ES aimed at optimizing f }
3. $\theta_h^t \leftarrow$ InitializationCMA() { auxiliary CMA-ES aimed at optimizing h_t }
4. fill θ_f^t with corresponding parameters stored in mean of distribution $\theta_h^t.m$
5. $\theta_f^{t+1} \leftarrow$ GenerationCMA(f, θ_f^t)
6. $t \leftarrow t + 1$
7. **repeat**
8. $\theta_f^{t+1} \leftarrow$ GenerationCMA(f, θ_f^t)
9. $\theta_h^{t+1} \leftarrow$ GenerationCMA(h_t, θ_h^t)
10.   fill θ_f^{t+1} with corresponding parameters stored in mean of distribution $\theta_h^{t+1}.m$
11. $t \leftarrow t + 1$
12. **until** stopping criterion is met

Specifically, while the primary CMA-ES optimizes $f(\boldsymbol{x})$ (line 8), the auxiliary CMA-ES maximizes $h_t(\theta)$ (line 9) by sampling and evaluating λ_h variants of θ_f^t. The updated mean of the auxiliary CMA-ES in the hyper-parameter space is used as a local estimate of the optimal hyper-parameter vector for the primary CMA-ES. Note that the auxiliary CMA-ES achieves a single iteration in the hyper-parameter space of the primary CMA-ES, with two motivations: limiting the computational cost of self-CMA-ES (which scales as λ_h times the time complexity of the CMA-ES), and preventing θ_f^t from overfitting the current sample $\boldsymbol{x}_{i:\lambda}^t, i = 1 \ldots \mu$.

Algorithm 3. Objective function $h_t(\theta)$

---
1. **Input**: $\theta, \theta_f^{t+1}, \theta_f^{t-1}, \theta_f^t, \mu, w_{sel,i}$ for $i = 1, \ldots, \mu$
2. $\theta_f^{'t-1} \leftarrow \theta$
3. $\theta_f^{'t} \leftarrow$ ReproduceGenerationCMA($f, \theta_f^{'t-1}$) using already evaluated $\theta_f^t.\boldsymbol{x}_{i:\lambda}$
4. $d_i \leftarrow \left\| \theta_f^{'t}.\sqrt{C^{-1}} \cdot (\theta_f^{t+1}.\boldsymbol{x}_i - \theta_f^{'t}.m) \right\|$; for $i = 1, \ldots, \theta_f^{t+1}.\lambda$
5. $p_i \leftarrow$ rank of $d_i, i = 1 \ldots \lambda$ sorted in decreasing order
6. $h(\theta) \leftarrow \sum_{i=1}^{\mu} w_{sel,i} p_{i:\lambda}$ { $i : \lambda$ denotes the rank of $\theta^{t+1}.\boldsymbol{x}_i$ }
7. **Output**: $h(\theta)$

---

4   Experimental Validation

The experimental validation of self-CMA-ES investigates the performance of the algorithm comparatively to CMA-ES on the BBOB noiseless problems [4]. Both algorithms are launched in IPOP scenario of restarts when the CMA-ES is restarted with doubled population size once stopping criteria [3] are met[2].

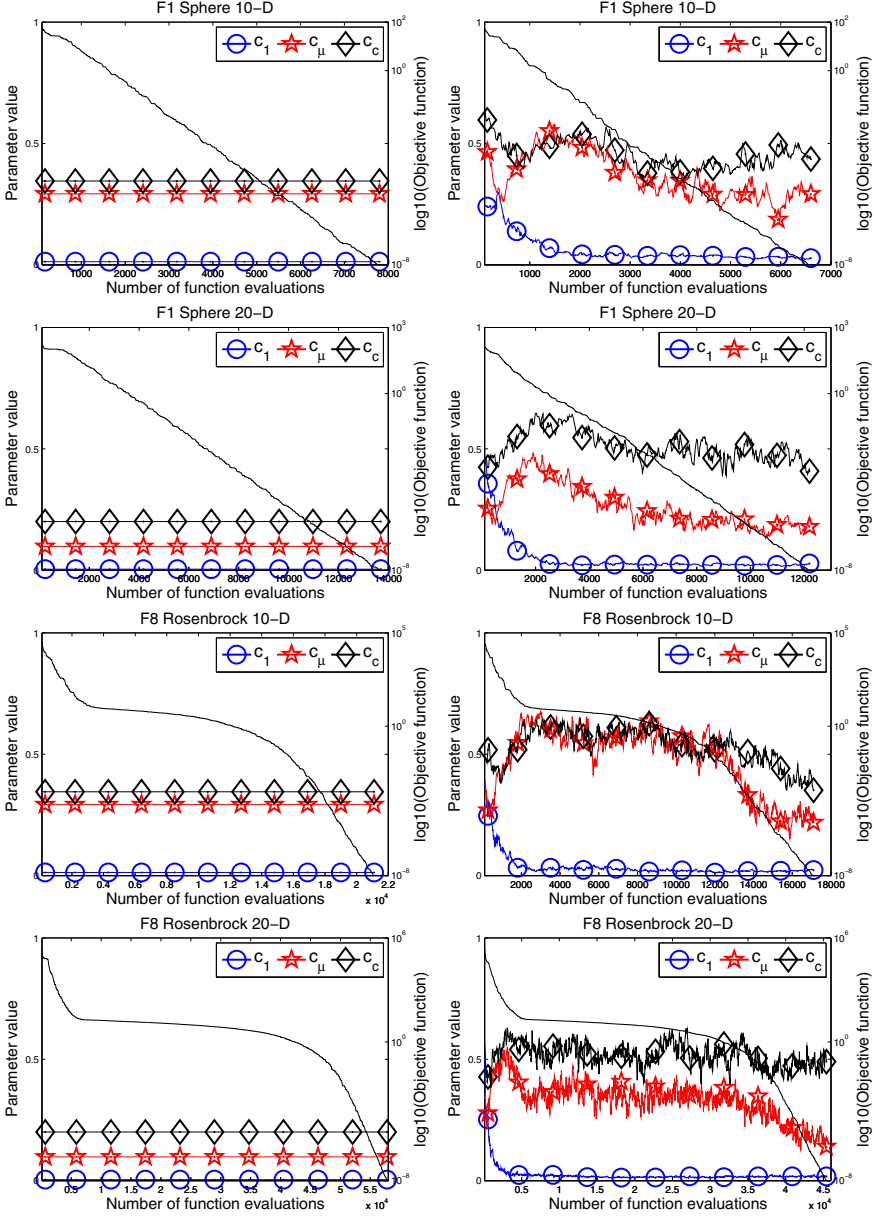[2] For the sake of reproducibility, the source code is available at
https://sites.google.com/site/selfcmappsn/

Fig. 1. Evolution of learning rates c_1, c_μ, c_c (lines with markers, left y-axis) and log10(objective function) (plain line, right y-axis) of CMA-ES (left column) and self-CMA-ES (right column) on 10- and 20-dimensional Sphere and Rosenbrock functions from [4]. The medians of 15 runs are shown.
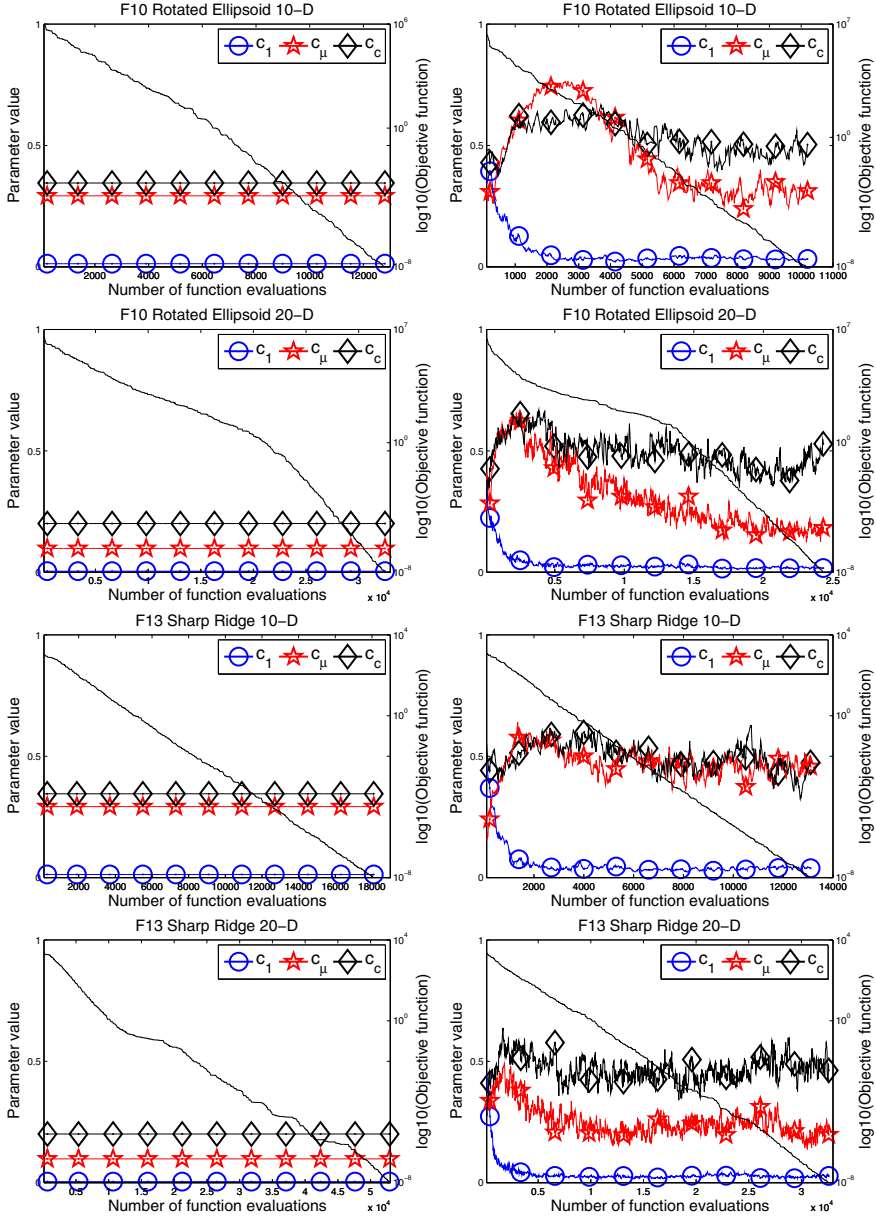
Fig. 2. Evolution of learning rates c_1, c_μ, c_c (lines with markers, left y-axis) and log10(objective function) (plain line, right y-axis) of CMA-ES (left column) and self-CMA-ES (right column) on 10- and 20-dimensional Rotated Ellipsoid and Sharp Ridge functions from [4]. The medians of 15 runs are shown.

The population size λ is chosen to be 100 for both CMA-ES and self-CMA-ES. We choose this value (about 10 times larger than the default one, see the default parameters of CMA-ES in Algorithm 1) to investigate how sub-optimal the other CMA-ES hyper-parameters, derived from λ, are in such a case, and whether self-CMA-ES can recover from this sub-optimality.

The auxiliary CMA-ES is concerned with optimizing hyper-parameters c_1, c_μ and c_c (Algorithm 1), responsible for the adaptation of the covariance matrix of the primary CMA-ES. These parameters range in $[0, .9]$ subject to $0 \leq c_1 + c_\mu \leq 0.9$; the constraint is meant to enforce a feasible $\boldsymbol{C}$ update for the primary CMA-ES (the decay factor of $\boldsymbol{C}$ should be in $[0, 1]$). Infeasible hyper-parameter solutions get a very large penalty, multiplied by the sum of distances of infeasible hyper-parameters to the range of feasibility.

We set $w_{sel,i} = 1/\mu$ for $i = 1, \ldots, \mu$ and $\mu = \lfloor \lambda/2 \rfloor$ to 50. The internal computational complexity of self-CMA-ES thus is $\lambda_h = 20$ times larger than the one of CMA-ES without lazy update (being reminded that the internal time complexity is usually negligible compared to the cost per objective function evaluation).

4.1   Results

Figures 1 and 2 display the comparative performances of CMA-ES (left) and self-CMA-ES (right) on 10 and 20-dimensional Sphere, Rosenbrock, Rotated Ellipsoid and Sharp ridge functions from the noiseless BBOB testbed [4] (medians out of 15 runs). Each plot shows the value of the hyper-parameters (left y-axis) together with the objective function (in logarithmic scale, right y-axis). Hyperparameters c_1, c_μ and c_c are constant and set to their default values for CMA-ES while they are adapted along evolution for self-CMA-ES.

In self-CMA-ES, the hyper-parameters are uniformly initialized in $[0, 0.9]$ (therefore the medians are close to 0.45) and they gradually converge to values which are estimated to provide the best update of the covariance matrix w.r.t. the ability to generate the current best individuals. It is seen that these values are problem and dimension-dependent. The values of c_1 are always much smaller than c_μ but are comparable to the default c_1. The values of c_μ and c_c and c_1 are almost always larger than the default ones; this is not a surprise for c_1 and c_μ, as their original default values are chosen in a rather conservative way to prevent degeneration of the covariance matrix.

Several interesting observations can be made about the dynamics of the parameter values. The value of c_μ is high most of the times on the Rosenbrock functions, but it decreases toward values similar to those of the Sphere functions, when close to the optimum. This effect is observed on most problems; indeed, on most problems fast adaptation of the covariance matrix will improve the performance in the beginning, while the distribution shape should remain stable when the covariance matrix is learned close to the optimum.

The overall performance of self-CMA-ES on the considered problems is comparable to that of CMA-ES, with a speed-up of a factor up to 1.5 on Sharp Ridge functions. The main result is the ability of self-CMA-ES to achieve the

online adaptation of the hyper-parameters depending on the problem at hand, side-stepping the use of long calibrated default settings[3].

4.2   Discussion

self-CMA-ES offers a proof of concept for the online adaptation of three CMA-ES hyper-parameters in terms of feasibility and usefulness. Previous studies on parameter settings for CMA-ES mostly considered offline tuning (see, e.g., [16,11]) and theoretical analysis dated back to the first papers on Evolution Strategies. The main limitation of these studies is that the suggested hyper-parameter values are usually specific to the (class of) analyzed problems. Furthermore, the suggested values are fixed, assuming that optimal parameter values remain constant along evolution. However, when optimizing a function whose landscape gradually changes when approaching the optimum, one may expect optimal hyper-parameter values to reflect this change as well.

Studies on the online adaptation of hyper-parameters (apart from σ, $\boldsymbol{m}$ and $\boldsymbol{C}$) usually consider population size in noisy [2], multi-modal [1,12] or expensive [9] optimization. A more closely related approach was proposed in [15] where the learning rate for step-size adaptation is adapted in a stochastic way similarly to Rprop-updates [10].

5   Conclusion and Perspectives

This paper proposes a principled approach for the self-adaptation of CMA-ES hyper-parameters, tackled as an auxiliary optimization problem: maximizing the likelihood of generating the best sampled solutions. The experimental validation of self-CMA-ES shows that the learning rates involved in the covariance matrix adaptation can be efficiently adapted on-line, with comparable or better results than CMA-ES. It is worth emphasizing that matching the performance of CMA-ES, the default setting of which represent a historical consensus between theoretical analysis and offline tuning, is nothing easy.

The main novelty of the paper is to offer an intrinsic assessment of the algorithm internal state, based on retrospective reasoning (given the best current solutions, how could the generation of these solutions have been made easier) and on one assumption (the optimal hyper-parameter values at time t are "sufficiently good" at time $t + 1$). Further work will investigate how this intrinsic assessment can support the self-adaptation of other continuous and discrete hyper-parameters used to deal with noisy, multi-modal and constrained optimization problems.

---

[3] $c_c = \frac{4}{n+4}$, $c_1 = \frac{2}{(n+1.3)^2 + \mu_w}$, $c_\mu = \frac{2\,(\mu_w - 2 + 1/\mu_w)}{(n+2)^2 + \mu_w}$.

References

1. Auger, A., Hansen, N.: A Restart CMA Evolution Strategy With Increasing Population Size. In: IEEE Congress on Evolutionary Computation, pp. 1769–1776. IEEE Press (2005)
2. Beyer, H.-G., Hellwig, M.: Controlling population size and mutation strength by meta-es under fitness noise. In: Proceedings of the Twelfth Workshop on Foundations of Genetic Algorithms XII, FOGA XII 2013, pp. 11–24. ACM (2013)
3. Hansen, N.: Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In: GECCO Companion, pp. 2389–2396 (2009)
4. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-Parameter Black-Box Optimization Benchmarking 2010: Experimental Setup. Technical report, INRIA (2010)
5. Hansen, N., Müller, S., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evolutionary Computation 11(1), 1–18 (2003)
6. Hansen, N., Ostermeier, A.: Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation. In: International Conference on Evolutionary Computation, pp. 312–317 (1996)
7. Hansen, N., Ostermeier, A.: Completely Derandomized Self-Adaptation in Evolution Strategies. Evolutionary Computation 9(2), 159–195 (2001)
8. Hansen, N., Ros, R.: Benchmarking a weighted negative covariance matrix update on the BBOB-2010 noisy testbed. In: GECCO 2010: Proceedings of the 12th Annual Conference Comp on Genetic and Evolutionary Computation, pp. 1681–1688. ACM, New York (2010)
9. Hoffmann, F., Holemann, S.: Controlled Model Assisted Evolution Strategy with Adaptive Preselection. In: International Symposium on Evolving Fuzzy Systems, pp. 182–187. IEEE (2006)
10. Igel, C., Hüsken, M.: Empirical evaluation of the improved rprop learning algorithms. Neurocomputing 50, 105–123 (2003)
11. Liao, T., Stützle, T.: Benchmark results for a simple hybrid algorithm on the CEC 2013 benchmark set for real-parameter optimization. In: IEEE Congress on Evolutionary Computation (CEC), pp. 1938–1944. IEEE Press (2013)
12. Loshchilov, I., Schoenauer, M., Sebag, M.: Alternative Restart Strategies for CMA-ES. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part I. LNCS, vol. 7491, pp. 296–305. Springer, Heidelberg (2012)
13. Loshchilov, I., Schoenauer, M., Sebag, M.: Self-Adaptive Surrogate-Assisted Covariance Matrix Adaptation Evolution Strategy. In: Genetic and Evolutionary Computation Conference (GECCO), pp. 321–328. ACM Press (July 2012)
14. Loshchilov, I., Schoenauer, M., Sebag, M.: Intensive Surrogate Model Exploitation in Self-adaptive Surrogate-assisted CMA-ES (saACM-ES). In: Genetic and Evolutionary Computation Conference, pp. 439–446. ACM (2013)
15. Schaul, T.: Comparing natural evolution strategies to bipop-cma-es on noiseless and noisy black-box optimization testbeds. In: Genetic and Evolutionary Computation Conference Companion, pp. 237–244. ACM (2012)
16. Smit, S., Eiben, A.: Beating the 'world champion' Evolutionary Algorithm via REVAC Tuning. IEEE Congress on Evolutionary Computation, 1–8 (2010)