Noisy Optimization by Evolution Strategies With Online Population Size Learning

Zhenhua Li, *Member, IEEE*, Shuo Zhang, Xinye Cai, *Member, IEEE*, Qingfu Zhang, *Fellow, IEEE*, Xiaomin Zhu, *Member, IEEE*, Zhun Fan, *Senior Member, IEEE*, and Xiuyi Jia, *Member, IEEE*

***Abstract*—Optimization modeling of real-world application problems usually involves noise from various sources. Noisy optimization imposes challenges to optimization methods since the objective values can be different for multiple evaluations. In this article, we propose a novel online population size learning (OPL) technique of evolution strategies for handling noisy optimization problems. By re-evaluating a fraction of the candidates, we measure the strength of noise level of the re-evaluated candidate solutions and adapt the population size according to the noise level. The proposed OPL combines the advantages of both explicit averaging by re-evaluations and the implicit averaging by large population size and overcomes their limitations. We incorporate it with the covariance matrix adaptation evolution strategy (CMA-ES) and obtain OPL-CMA-ES. Compared with the existing noise handling technique, the proposed OPL is much simpler in both concepts and computation. We conduct comprehensive experiments to evaluate the algorithm's performance on standard problems with Gaussian noise. We further evaluate the performance of OPL-CMA-ES on the black-box optimization benchmarks (BBOBs) noisy testbed, which is a standard platform for comparing black-box optimization algorithms, compared with the state-of-the-art noise-handling algorithms. The experimental results show that OPL-CMA-ES achieves remarkable performance and outperforms the compared variants.**

***Index Terms*—Evolution strategies (ESs), noise strength, noisy optimization, online population size learning (OPL).**

Manuscript received July 1, 2021; accepted November 16, 2021. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62102178, Grant 62072234, Grant 61732006, and Grant 61773208; in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20200443 and Grant BK20181288; in part by the China Postdoctoral Science Foundation under Grant 2015M571751; and in part by the Fundamental Research Funds for the Central Universities under Grant NS2017070. This article was recommended by Associate Editor Y. Dong. *(Corresponding author: Xinye Cai.)*

Zhenhua Li, Shuo Zhang, and Xinye Cai are with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, Jiangsu, China, and also with the Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210023, China (e-mail: zhenhua.li@nuaa.edu.cn; zhangshuo.nuaa@foxmail.com; xinye@nuaa.edu.cn).

Qingfu Zhang is with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: qingfu.zhang@cityu.edu.hk).

Xiaomin Zhu is with the College of Systems Engineering, National University of Defense Technology, Changsha 410073, China (e-mail: xmzhu@nudt.edu.cn).

Zhun Fan is with the Department of Electronic Engineering, Shantou University, Shantou 515063, Guangdong, China (e-mail: zfan@stu.edu.cn).

Xiuyi Jia is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: jiaxy@njust.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TSMC.2021.3131482.

Digital Object Identifier 10.1109/TSMC.2021.3131482

I. Introduction

MANY real-world applications are formulated for optimization problems. During the modeling, noise stems from various sources and leads to noisy optimization problems (NOPs) [1], [2]. In the presence of noise, the objective values of a specific point with multiple evaluations can be different. The noisy optimization imposes important challenges for optimization algorithms that assume the exact objective value for a given point. Evolutionary algorithms (EAs) repeatedly sample a population of candidate solutions and select the most promising ones to generate highly qualified candidate solutions at each generation. EAs are more robust against noise [3]–[5] and have attracted an increasing amount of attention in the presence of noise [6], [7]. The selection of EAs depends on the comparison of objective values. The selection can be distorted when the noise level becomes strong enough, and the evolutionary search is misled and fails. To enhance the performance of EAs under the noise environment, a straightforward approach is to evaluate each candidate solution multiple times and average the fitness [8]. Re-evaluating the candidate solutions can effectively reduce the variance of noise and one can obtain a more accurate objective value. One of the successful re-evaluation methods is the uncertainty handling (UH) technique for evolution strategies (UH-CMA-ES). However, empirical results illustrate that the re-revaluation and explicit average methods can perform well only on the problems with Gaussian noise of low to medium level. For non-Gaussian noise, such as uniform noise and Cauchy noise, the explicit averaging methods UH-CMA-ES cannot perform well.

Another method is to use a large population size. As EAs repeatedly sample the promising area, the sampled candidate solutions are usually highly similar, and the effect of noise in evaluating each candidate solution can be largely compensated by the similar ones. This implicit averaging reduces the effects of noise and improves the performance of EAs for NOPs. To reduce the number of function evaluations (FEs), the population size is usually adapted based on the estimated noise level during the optimization procedure [9]–[11]. The noise level can be estimated based on the reduction of the noisy objective values, or the accuracy of the update mechanisms. These methods cannot take advantage of re-evaluations and converge

slower than the re-evaluation methods in the Gaussian noise case.

In this article, we propose a new online population size learning (OPL) method for NOPs. We incorporate OPL with the covariance matrix adaptation (CMA-ES) as the basic search engine, which is one of the most successful variants of EAs, and denote the obtained algorithm by OPL-CMA-ES. The main features of OPL-CMA-ES are as follows.

1) OPL combines the advantages of both explicit averaging and implicit averaging methods and overcomes the limitations of both methods. We re-evaluate a small fraction of the whole population of candidate solutions and measure the noise level by the rank changes of the re-evaluated solutions. The noise level is then used to adapt the population size. By re-evaluating a fraction of the candidate solutions, OPL reduces the effects of noise by following the explicit averaging and adapting the population size improves the robustness on more general NOPs.
2) OPL represents a general framework for handling NOPs. It does not depend on any specific evolutionary operations and adaption mechanisms and is well suited for any ranking-based EAs. We incorporate it with the covariance matrix adaptation as the search engine and present OPL-CMA-ES.

We conduct comprehensive experiments to evaluate the performance of the proposed OPL-CMA-ES, compared with state-of-the-art noisy evolutionary optimization algorithms. The experimental results show that OPL-CMA-ES achieves superior performance in terms of the number of FEs to reach the predefined accuracy and the ratio of solved functions. We evaluate its performance on the standard black-box optimization benchmark (BBOB) noisy testbed and validate that OPL-CMA-ES achieves STOA performance. In particular, OPL-CMA-ES outperforms all the compared algorithms and shows more promising performance as the dimension increases.

The remainder of this article is organized as follows. Section II presents the problem setting of noisy optimization in the continuous domain and summarizes the noise handling methods. In Section III, we introduce the proposed OPL-CMA-ES in detail. We study the performance of OPL-CMA-ES on some basic noisy test functions in Section IV. In Section V, we further comprehensively evaluate the performance of OPL-CMA-ES on the BBOB noisy testbed, compared with other existing noise handling algorithms. Finally, we conclude this work and present our future work in Section VI.

II. Background and Related Works

In this section, we present the problem setting of noisy optimization, and the noise handling techniques for EAs.

A. Noisy Optimization Problems

In this article, we consider unconstrained NOPs in the continuous domain

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad \widetilde{f}(\mathbf{x}) = f(\mathbf{x}) + \delta \tag{1}$$

where $f(\mathbf{x})$ is the true objective function part and δ is a stochastic part. The random variate δ can scale with the objective function $\delta = \epsilon \cdot f(\mathbf{x})$, or not as $\delta = \epsilon$, where ϵ is the noise distributed according to some unknown distribution. The goal of noisy optimization is to find the optimum for the expected objective function

$$J(\mathbf{x}) = \mathbb{E}\big[\widetilde{f}(\mathbf{x})\big]. \tag{2}$$

Under the assumption that the random number δ is unbiased, we have $\mathbb{E}[\widetilde{f}(\mathbf{x})] = f(\mathbf{x})$. In practice, we cannot directly evaluate $J(\mathbf{x})$. Instead, we need to approximate the expected objective function $J(\mathbf{x})$ by evaluating the objective function $\widetilde{f}(\mathbf{x})$ multiple times for each candidate solution $\mathbf{x}$, namely

$$\hat{J}(\mathbf{x}) = f(\mathbf{x}) + \frac{1}{N_{\text{eval}}} \sum_{i=1}^{N_{\text{eval}}} \delta_i \tag{3}$$

where N_{eval} is the number of FE times for each candidate solution, and δ_i for $i = 1, \ldots, N_{\text{eval}}$ are independent identically distributed copies of δ. In the noise-free case, N_{eval} is set to 1, i.e., each candidate solution is evaluated only once.

B. Noise Handling Techniques in Evolutionary Algorithms

The population nature of EAs makes them more robust and suitable for NOPs. As the FEs are computationally expensive and regarded as the cost of optimization, the goal of EAs for NOPs is to find more accurate solutions with as few FEs as possible. We summarize some existing noise handling methods in EAs in the following section and refer to [3] for a more comprehensive survey of this topic.

1) Explicit Averaging With Re-Evaluations: Explicit averaging methods are straightforward to reduce the noise level by evaluating the candidate solutions multiple times and averaging the objective values. For an additive Gaussian noise, re-evaluating each candidate N_{eval} times can effectively reduce the noise level of objective function $\widetilde{f}(\mathbf{x})$ by a factor of $\sqrt{N_{\text{eval}}}$. In practice, evaluating the objective function multiple times increases the number of FEs.

To save the number of FEs, researchers propose to adapt the number of re-evaluations. The optimal computing budget allocation scheme assigns the optimal re-evaluations to each candidate solution to reduce the influence of noise [12], [13]. The UH technique [14] adapts the times of re-evaluations N_{eval} for CMA-ES based on the estimated noise level, leading to the UH-CMA-ES. However, the effect of noise can only be reduced by re-evaluation if the noise is not heavy-tailed [15]. Hence, the explicit averaging methods for handling noise can hardly be regarded as a general method. In practice, the explicit averaging methods are well suited for the noisy objective functions with Gaussian noise of low to medium level but do not perform well on the noisy objective functions with non-Gaussian noise.

2) Implicit Averaging With Large Population Size: Using a large population size can be considered as an implicit average approach for solving NOPs [16]. EAs repeatedly generate many similar solutions in consecutive generations when searching the promising areas, the effect of noise in the objective values is very highly to be implicitly compensated by that of similar candidate solutions with a large population.

Regardless of the type of noise, increasing the population size instead of explicit averaging can reduce the effects of noise.

To reduce the computational cost, the population size is usually adapted during the optimization procedure [9]–[11]. In pcCMSA-ES [9], the population size is adapted by analyzing the dynamics of the objective values. The CMAES-APOP [17] adjusts the population size according to the improvements of the objective value over generations. The PSAaLmC [10] adjusts the population size according to the accuracy of the update for the distribution parameters, following the natural gradient from the structure of information-geometric optimization [18]. The idea is further extended to PSA-CMA-ES [19] following the paradigm of the standard CMA-ES. It adjusts the population size according to the accuracy of the updates. These methods have the following important drawbacks. First, these methods usually converge slower than the re-evaluation methods on the Gaussian noisy problems as they do not make use of the re-evaluations. Second, the adaption of population size usually depends on some specific mechanisms and can be hardly generalized to other variants of evolution strategies (ESs) and EAs. Third, these methods are usually very complicated and not easy to implement.

3) Robust Selection: The robust selection scheme revises the selection operation to enhance the reliability of quality solutions from the population [20], [21]. The threshold-based selection uses a threshold for selection and accepts an offspring individual if its fitness is better than that of the parent by a threshold [22]–[24]. However, the performance of threshold-based selection depends intensely on the sensible predefined threshold, which depends on the optimization problems and is usually hard to design.

4) Hybrid Search: Some researchers view the NOPs as noise-induced multimodal problems [25] and propose to handle them by hybridizing evolutionary operators to enhance the global search ability and robustness in the presence of noise [26]. DE-PSO [27] is an improved particle swarm optimization algorithm (PSO) that applies the differential variation mechanism employed in the differential evolution algorithm (DE) to adapt the velocity of particles. The multiple offspring sampling (MOS) [28] is a hybrid algorithm that uses MOS structures to combine the different EAs to enhance robustness. The main drawback of these algorithms is that they fail to take advantage of the specific information of noisy objective values and usually converge slowly.

III. OPL-CMA-ES Algorithm

In this article, we propose a novel OPL method for ESs to handle NOPs. We adapt the population size depending on the detected noise level of the uncertain environment. Algorithm 1 presents the general framework of the proposed OPL for ESs. In the following sections, we will introduce the involved components of our algorithm in detail.

A. Noise Measurement

ESs select the candidate solutions by comparing their objective values. Due to the ranking-based selection in ESs, noise affects the selection procedure when the rankings among the candidate solutions change. We measure the level of noise strength by the rank changes of the multiple evaluations of specific candidate solutions following the idea of [14] and then adapt the population size by the normalized noise strength.

Algorithm 1 General Framework of OPL for ESs

1: Initialize λ_0 and search distribution $\mathcal{N}(\mathbf{m}_0, \sigma_0^2\mathbf{C}_0)$.
2: **repeat**
3: Generate λ_t candidate solutions $\mathbf{x}_i \sim \mathcal{N}(\boldsymbol{m}_t, \sigma_t^2\mathbf{C}_t)$.
4: Detect the noise level by re-evaluating a fraction of the candidate solutions (Section III-A).
5: Adapt the population size λ_t (Section III-B).
6: Update the distribution parameters $\mathbf{m}_t, \sigma_t, \mathbf{C}_t$ (Section III-C).
7: Update the parameters depending on the population size λ_t.
8: $t = t + 1$
9: **until** stopping criterion is met

1) Rank Changes of Re-Evaluated Solutions: Fig. 1 presents the rank changes of the candidate solutions with some of them re-evaluated. In this case, we consider an additive Gaussian noise $\delta \sim \mathcal{N}(0, \sigma_\epsilon^2)$ with standard deviation σ_ϵ. In Fig. 1, we present $f(x)$ and a standard deviation $f(x) \pm \sigma_\epsilon$ with different level of noise strengths $\sigma_\epsilon^{(1)} < \sigma_\epsilon^{(2)} < \sigma_\epsilon^{(3)}$. The black points mark the objective values of candidate solutions, and the blue triangles mark the re-evaluated objective values of the first three candidate solutions. We have the following observations and discussions according to Fig. 1.

1) Fig. 1(a) shows that the rankings of candidate solutions $x_i, i = 1, 2, 3$, do not change after the re-evaluation. It shows that the selection is not affected by the noise. It indicates that the noise strength is weak.
2) Fig. 1(b) shows that the rankings of candidate solutions after re-evaluation change from $\langle 3, 2, 1, 4, 5, 6\rangle$ to $\langle 2, 3, 1, 4, 5, 6\rangle$. Hence, with the medium level of noise strength, the selection is slightly affected by the noise.
3) Fig. 1(c) shows the changes in objective values of re-evaluated solutions. The rankings of candidate solutions change from $\langle 3, 2, 1, 4, 5, 6\rangle$ to $\langle 1, 4, 2, 3, 5, 6\rangle$ after the re-evaluation. The dramatic change in the ranking indicates that the level of noise strength is strong and the selection is seriously affected by the noise.

The above observations validate that the rank changes of re-evaluated solutions can be used as a measure of the noise strength.

2) Noise Strength Measurement: We measure the noise strength by the changes in the rank of the re-evaluated solutions. The procedure of noise strength level detection is revised from the noise measurement of [14]. The steps of detecting the noise strength are presented in Algorithm 2.

1) Compute the number of re-evaluated candidate solutions by means of $\lambda_{\text{reev}} = \lfloor r_\lambda \cdot \lambda \rfloor$, where $r_\lambda \leq 1$ is the ratio of re-evaluation (line 2).
2) Re-evaluate the first λ_{reev} solutions and let $\mathcal{L} = \{\widetilde{f}_1, \ldots, \widetilde{f}_\lambda, \widetilde{f}_1^{\text{new}}, \ldots, \widetilde{f}_{\lambda_{\text{reev}}}^{\text{new}}\}$ (lines 3 and 4).

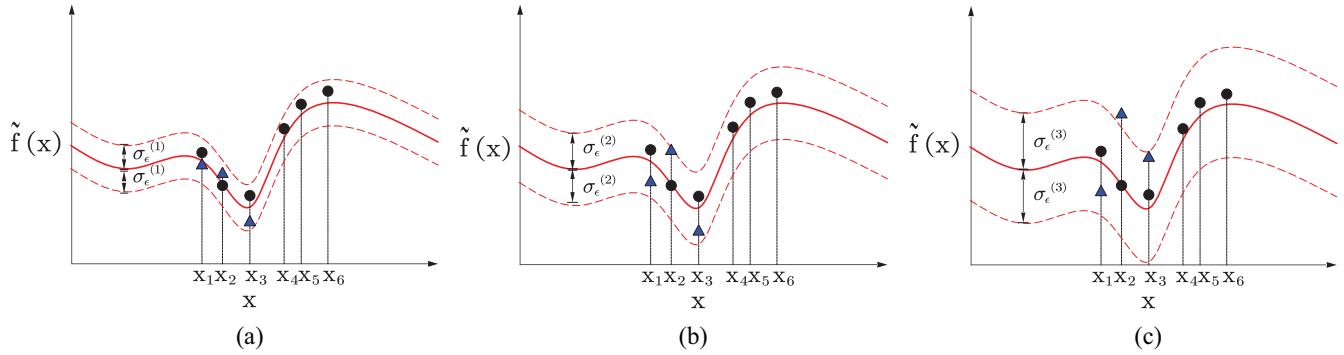

Fig. 1. Illustration of rank changes within candidate solutions at an iteration, minimizing an objective function $\widehat{f}$ with a 1-D continuous domain. The figure shows three rank changes within candidate solutions from different noise situations. Each subfigure shows: noise-free objective function f (solid red line); a standard deviation for f, $f \pm \sigma_\epsilon$ (dashed red lines); objective function $\widetilde{f}$ values of candidate solutions (black points); and re-evaluated objective function values of the first three candidate solutions (blue triangles). (a) Weak. (b) Moderate. (c) Severe.

Algorithm 2 Detection($[\mathbf{x}_1 \cdots, \mathbf{x}_\lambda]$, $[\widetilde{f}_1, \cdots, \widetilde{f}_\lambda]$)

1: **Initialize:** $r_\lambda = \max(0.1, 2/\lambda)$, $\theta = 0.2$
2: $\lambda_{\text{reev}} = \lfloor r_\lambda \cdot \lambda \rfloor$
3: **for** $i = 1, \cdots, \lambda_{\text{reev}}$ **do** $\widetilde{f}_i^{\text{new}} \leftarrow \widetilde{f}(\mathbf{x}_i)$
4: $\mathcal{L} = \{\widetilde{f}_1, \cdots, \widetilde{f}_\lambda, \widetilde{f}_1^{\text{new}}, \cdots, \widetilde{f}_{\lambda_{\text{reev}}}^{\text{new}}\}$
5: **for** $i = 1, \cdots, \lambda_{\text{reev}}$ **do**
6: $\quad \Delta_i = \text{rank}(\widetilde{f}_i^{\text{new}}) - \text{rank}(\widetilde{f}_i) - \text{sign}(\text{rank}(\widetilde{f}_i^{\text{new}}) - \text{rank}(\widetilde{f}_i))$
7: **end for**
8: $r = \frac{1}{\lambda_{\text{reev}} \cdot \lambda} \sum_{i=1}^{\lambda_{\text{reev}}} \big(2|\Delta_i| - \Delta_\theta(\text{rank}(\widetilde{f}_i^{\text{new}}) - \mathbb{1}\{\widetilde{f}_i^{\text{new}} > \widetilde{f}_i\}) - \Delta_\theta(\text{rank}(\widetilde{f}_i) - \mathbb{1}\{\widetilde{f}_i > \widetilde{f}_i^{\text{new}}\})\big)$
9: **for** $i = 1, \cdots, \lambda_{\text{reev}}$ **do** $\widetilde{f}_i \leftarrow \frac{1}{2}(\widetilde{f}_i + \widetilde{f}_i^{\text{new}})$
10: **return** r, $[\widetilde{f}_1, \cdots, \widetilde{f}_\lambda]$

3) Calculate the rank change of each re-evaluated solution Δ_i (lines 5–7), where $\text{rank}(\widetilde{f}_i)$ is the rank of $\widetilde{f}_i$ in the set $\mathcal{L}$. The rank change, $|\Delta_i| \in \{0, 1, \ldots, \lambda + \lambda_{\text{reev}} - 2\}$, counts the number of objective values from the set $\mathcal{L} \backslash \{\widetilde{f}_i, \widetilde{f}_i^{\text{new}}\}$ that lie between $\widetilde{f}_i$ and $\widetilde{f}_i^{\text{new}}$. The sign function sign() will return $+1$ if its argument is positive, or -1 if its argument is negative, or 0 otherwise.
4) Compute the normalized noise strength r (line 8) with respect to a threshold θ. The rank change Δ_i is compared with $\Delta_\theta(R)$, which denotes the $\theta/2$ percentile of the possible rank changes $|1-R|, |2-R|, \ldots, |\lambda+\lambda_{\text{reev}}-1-R|$ when having the original rank R. The indicator function $\mathbb{1}$ will return $+1$ if its argument is true and 0 otherwise. Then, it is averaged over the λ_{reev} re-evaluated solutions and normalized by a factor $1/\lambda$ into $(-1, 1)$.
5) Average the objective values of the re-evaluated candidate solutions for the selection procedure of CMA-ES (line 9).

The normalized rank changes r measure the noise strength of the noisy objective function. If $r > 0$, the rank changes of re-evaluated solutions are larger than the acceptance threshold. In this case, the objective values and selection are dramatically affected by the noise.

In this algorithm, two parameters are involved in the calculation of r.

1) r_λ: The re-evaluation ratio $r_\lambda \leq 1$ controls the fraction of candidate solutions to be re-evaluated. It should be large enough for sufficient reliable noise measurement and as small as possible for optimization speed. Generally, we use $r_\lambda = \max(0.1, 2/\lambda)$ following the setting of [14], i.e., we re-evaluate on average $\max(\lambda/10, 2)$ candidate solutions at each iteration.
2) θ: The parameter $\theta \in [0, 1]$ controls the acceptance threshold for the measured rank changes. It denotes the noise level that we can tolerate. The noise level that we can tolerate decreases as the threshold θ decreases. In this article, we generally set $\theta = 0.2$ as [14] unless specified otherwise.

These parameters are well evaluated for the UH methods. We evaluate that the proposed algorithm's performance is relatively robust to the parameter settings.

B. Online Population Size Learning

We learn the population size according to the measured noise strength r. After the normalization, we accumulate r over iterations to reduce the influences of randomness

$$\psi_{t+1} = (1 - c_\lambda)\psi_t + c_\lambda r \tag{4}$$

where $c_\lambda > 0$ is a constant changing rate. Since the accumulated r is in the range $(-1, 1)$, we set $c_\lambda = 0.2$. Then, the population size is adapted as

$$\lambda_{t+1} = \lambda_t \cdot \exp\left(\frac{\psi_{t+1}}{d_\lambda}\right) \tag{5}$$

where d_λ is a damping parameter that scales the change magnitude of $\ln \lambda_t$. It can reduce the variance of the population size change, and a large d_λ can slow down the adaptation. In this article, we set $d_\lambda = 1 + c_\lambda$.

The population size is adapted by using the accumulated noise strength. If $r > 0$ over multiple generations, it implies the rank changes within candidate solutions are severe and ψ_{t+1} tends to be positive. Hence, the population size tends to increase. Otherwise, if $r < 0$ over generations, the rank changes are weak and $\psi_{t+1} < 0$, which means a decrease in the population size.

Then, we restrict the population size in the range of $[\lambda_{\min}, \lambda_{\max}]$. The lower bound $\lambda_{\min}$ is set to the default value

($\lambda_{\text{def}} = 4 + \lfloor 3 \ln n \rfloor$) of the CMA-ES, i.e., $\lambda_{\min} = \lambda_{\text{def}}$. Following the setting of [29], the upper bound $\lambda_{\max}$ is set to be $\lambda_{\max} = (20n + 30)\lambda_{\text{def}}$.

C. Update the Distribution Parameters

We use the covariance matrix adaptation evolution strategy (CMA-ES) as the basic search engine. Any variants of ESs, such as the natural ESs [30] and Cholesky variants [31], [32], can be used as the search engine without change of the algorithm framework. After the re-evaluation and detection of noise level, we sort the evaluated points according to their function values $\widetilde{f}(\mathbf{x}_{1:\lambda}) \leq \widetilde{f}(\mathbf{x}_{2:\lambda}) \leq \cdots \leq \widetilde{f}(\mathbf{x}_{\lambda:\lambda})$. The distribution mean is updated by a weighted sum of the best $\mu = \lfloor \lambda/2 \rfloor$ parent individuals selected among λ generated offspring individuals as

$$\mathbf{m}_{t+1} = \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda} \tag{6}$$

where $w_i > 0, i = 1, \ldots, \mu$, are the weights. An evolution path $\mathbf{p}$ is constructed by tracking the movements of the distribution mean $\mathbf{m}_t$ and update the covariance matrix

$$\mathbf{p}_{t+1} = (1 - c_c)\mathbf{p}_t + \sqrt{c_c(2 - c_c)}\sqrt{\mu_w}\frac{\mathbf{m}_{t+1} - \mathbf{m}_t}{\sigma_t} \tag{7}$$

$$\mathbf{C}_{t+1} = \big(1 - c_1 - c_\mu\big)\mathbf{C}_t + c_1 \mathbf{p}_{t+1}\mathbf{p}_{t+1}^T + c_\mu \boldsymbol{C}_{MLE} \tag{8}$$

where $\mu_w = 1/\sum_{i=1}^{\mu} w_i^2$, $\boldsymbol{C}_{MLE} = (1/\sigma_t^2)\sum_{i=1}^{\mu} w_i(\mathbf{x}_{i:\lambda} - \mathbf{m}_t)(\mathbf{x}_{i:\lambda} - \mathbf{m}_t)^T$ is the weighted maximum-likelihood estimation of the covariance matrix of the selected points, and c_c, c_1, and c_μ are learning rates. In practice, these learning rates are set $c_c^{-1} \approx n$ and $c_1^{-1}, c_\mu^{-1} \approx n^2$, i.e., inversely proportional to the number of parameters to be adapted.

In addition to the covariance matrix, the step size is adapted by constructing an evolution path $\boldsymbol{s}$ and comparing its strength to the expectation $\mathbb{E}[\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|]$

$$\mathbf{s}_{t+1} = (1 - c_\sigma)\mathbf{s}_t + \sqrt{c_\sigma(2 - c_\sigma)}\sqrt{\mu_w}\mathbf{C}_t^{-\frac{1}{2}}\frac{\mathbf{m}_{t+1} - \mathbf{m}_t}{\sigma_t} \tag{9}$$

$$\sigma_{t+1} = \sigma_t \cdot \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\mathbf{s}_{t+1}\|}{\mathbb{E}[\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|]} - 1\right)\right) \tag{10}$$

where $\mathbf{C}_t^{-(1/2)}$ is the inversion of the unique symmetric factor, c_σ is the learning rate setting to $c_\sigma \approx (1/n)$, and $d_\sigma \approx 1$ is the damping parameter. We refer to [33] for more details and discussions on the updates and parameter settings.

D. OPL-CMA-ES

We term the resulting algorithm by OPL-CMA-ES, the CMA-ES with OPL. The algorithm framework and involved parameters are presented in Algorithm 3. In each generation, λ_t candidate solutions in the population are generated from the Gaussian distribution (lines 5–8). The rank changes are detected by re-evaluating a fraction of the candidate solutions (lines 9 and 10). In the online learning of population size, the cumulative noise strength ψ_{t+1} is first computed (line 12). Then, the population size is adapted according to the proposed mechanism and is restricted in the range of $[\lambda_{\min}, \lambda_{\max}]$ (lines 13 and 14). The distribution parameters

Algorithm 3 OPL-CMA-ES

1: **Input:** $\mathbf{m}_0 \in \mathbb{R}^n, \sigma_0 \in \mathbb{R}_+$
2: **Set:** $\lambda_{\text{def}} = 4 + \lfloor 3 \ln n \rfloor$, $\mu = \lfloor \lambda_{\text{def}}/2 \rfloor$, $c_c = \frac{4}{n+4}$, $w_i = \frac{\ln(\mu+0.5) - \ln i}{\sum_{k=1}^{\mu} \ln(\mu+0.5) - \ln k}$, $i = 1, \cdots, \mu$, $\mu_w = \frac{1}{\sum_{i=1}^{\mu} w_i^2}$, $c_1 = \frac{2}{(n+1.3)^2 + \mu_w}$, $c_\sigma = \frac{\mu_w + 2}{n + \mu_w + 3}$, $c_\mu = \frac{2(\mu_w - 2 + \frac{1}{\mu_w})}{(n+2)^2 + \mu_w}$, $d_\sigma = 1 + 2\max(0, \sqrt{\frac{\mu_w - 1}{n+1}} - 1) + c_\sigma$, $c_\lambda = 0.2$, $\lambda_{\text{def}} = \lambda$, $d_\lambda = 1 + c_\lambda$
3: **Initialize:** $\mathbf{C}_0 = \mathbf{I}$, $\mathbf{p}_0 = \mathbf{0}$, $\mathbf{s}_0 = \mathbf{0}$, $\lambda_0 = k_n \times \lambda_{\text{def}}$, $t = 0$, $\psi_0 = 0$
4: **repeat**
5: // sample λ_t candidate solutions from $\mathcal{N}(\boldsymbol{m}_t, \sigma_t^2 \boldsymbol{C}_t)$
6: **for** $i = 1$ to λ_t **do**
7: $\mathbf{x}_i = \mathbf{m}_t + \sigma_t \mathbf{y}_i$ with $\mathbf{y}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_t)$
8: **end for**
9: // compute the noise strength
10: $r, [\widetilde{f}_1, \cdots, \widetilde{f}_\lambda] \leftarrow$ **detection**$([\mathbf{x}_1 \cdots, \mathbf{x}_\lambda], [\widetilde{f}_1, \cdots, \widetilde{f}_\lambda])$
11: // update the population size
12: $\psi_{t+1} = (1 - c_\lambda)\psi_t + c_\lambda r$
13: $\lambda_{t+1} = \lambda_t \cdot \exp\left(\frac{\psi_{t+1}}{d_\lambda}\right)$
14: $\lambda_{t+1} = \min(\max(\lfloor \lambda_{t+1} \rfloor, \lambda_{\min}), \lambda_{\max})$
15: // update the distribution parameters
16: sort $\widetilde{f}(\mathbf{x}_{1:\lambda}) \leq \widetilde{f}(\mathbf{x}_{2:\lambda}) \leq \cdots \leq \widetilde{f}(\mathbf{x}_{\lambda:\lambda})$
17: $\mathbf{m}_{t+1} = \mathbf{m}_t + \sum_{i=1}^{\mu} w_i(\mathbf{x}_{i:\lambda} - \mathbf{m}_t)$
18: $\mathbf{p}_{t+1} = (1 - c_c)\mathbf{p}_t + \sqrt{c_c(2 - c_c)}\sqrt{\mu_w}\frac{\mathbf{m}_{t+1} - \mathbf{m}_t}{\sigma_t}$
19: $\mathbf{s}_{t+1} = (1 - c_\sigma)\mathbf{s}_t + \sqrt{c_\sigma(2 - c_\sigma)}\sqrt{\mu_w}\mathbf{C}_t^{-\frac{1}{2}}\frac{\mathbf{m}_{t+1} - \mathbf{m}_t}{\sigma_t}$
20: $\mathbf{C}_{t+1} = (1 - c_1 - c_\mu)\mathbf{C}_t + c_1 \mathbf{p}_{t+1}\mathbf{p}_{t+1}^T + c_\mu \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}\mathbf{y}_{i:\lambda}^T$
21: $\sigma_{t+1} = \sigma_t \cdot \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||\mathbf{s}_{t+1}||}{\mathbb{E}[||\mathcal{N}(\mathbf{0}, \mathbf{I})||]} - 1\right)\right)$
22: **if** λ_{t+1} is changed **then**
23: update the parameters depending on λ_{t+1}
24: **end if**
25: $t = t + 1$
26: **until** stopping criterion is met

$\mathbf{m}$, σ, and $\mathbf{C}$ are updated as the default CMA-ES (lines 15–21). At the end of each generation, the parameters that depend on the population size are accordingly updated (lines 22–24).

In OPL-CMA-ES, we do not correct the step size after the adaption of population size for simplicity. This differs from that of PSA-CMA-ES [19] that corrects the step size to make the population size adaptation stable [34]. Experimental results validate that the effect of the step size correction is marginal.

We use a large initial population size for NOPs. The initial population size is increased by a factor of $k_n = \min(4n, 100)$ to prevent the potential premature convergence.

IV. Experimental Studies

In this section, we evaluate the performance of OPL-CMA-ES on some basic noisy test functions defined in Table I. These test functions are characterized with different features and we can investigate the convergence behaviors of OPL-CMA-ES on

TABLE I
TEST PROBLEMS

$f_{\text{Sphere}}(\mathbf{x})$	$= \sum_{i=1}^{n} x_i^2$
$f_{\text{Benign Ellipsoid}}(\mathbf{x})$	$= \sum_{i=1}^{n} 10^{\frac{2(i-1)}{n-1}} x_i^2$
$f_{\text{Ellipsoid}}(\mathbf{x})$	$= \sum_{i=1}^{n} 10^{\frac{6(i-1)}{n-1}} x_i^2$
$f_{\text{Diffpow}}(\mathbf{x})$	$= \sum_{i=1}^{n} \lvert x\rvert_i^{2+\frac{10(i-1)}{n-1}}$
$f_{\text{Cigar}}(\mathbf{x})$	$= x_1^2 + 10^6 \cdot \sum_{i=2}^{n} x_i^2$
$f_{\text{Tablet}}(\mathbf{x})$	$= 10^6 \cdot x_1^2 + \sum_{i=2}^{n} x_i^2$
$f_{\text{Rosenbrock}}(\mathbf{x})$	$= \sum_{i=1}^{n-1} 100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2$
$f_{\text{Rastrigin}}(\mathbf{x})$	$= \sum_{i=1}^{n} (x_i^2 + 10(1 - \cos 2\pi x_i))$
$f_{\text{Ackley}}(\mathbf{x})$	$= -20 \exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) + 20$
	$- \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right) + \exp(1)$

different local landscapes. We consider the additive Gaussian noise $\widetilde{f}(x) = f(x) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ and strong noise strength $\sigma_\epsilon = 1$. In the experiments, the initial distribution mean $\mathbf{m}_0$ is randomly uniformly generated in the range $[1, 5]^n$, and $\sigma_0 = 2$. We set the maximum FEs to $10^5 \times n$. Each algorithm independently runs 51 times. We conduct the Wilcoxon rank-sum hypothesis test at significance level $p = 0.05$ to assess the differences.

We take CMA-ES with the default population size and noise handling methods into the comparison, including the pcCMSA-ES, CMAES-APOP, and UH-CMA-ES. Both pcCMSA-ES and CMAES-APOP handle NOPs by adapting the population size. The pcCMSA-ES controls the population size according to the analysis of the objective dynamics [9]. The CMAES-APOP adapts the population size by using the improvements of objective values. The UH-CMA-ES adapts the number of re-evaluations of candidate solutions to reduce the variance of the objective value [14].

A. Compared Algorithms

Fig. 2 presents the median run of each algorithm,[1] where the dashed line denotes the noise strength $\sigma_\epsilon = 1$. The main observations are as follows.

1) *CMA-ES:* OPL-CMA-ES outperforms CMA-ES dramatically on all the test functions. CMA-ES with the default population size stops improving the objective value once the objective value is comparative to the noise strength. With the same FEs, OPL-CMA-ES achieves much better final solutions than CMA-ES, which cannot find qualified solutions.
2) *UH-CMA-ES:* OPL-CMA-ES performs much better results than that of UH-CMA-ES on all the test functions.
3) *pcCMSA-ES:* OPL-CMA-ES achieves superior results than pcCMSA-ES on the test functions except for the Sphere and Diffpow function. On the Benign Ellipsoid and Ellipsoid function, pcCMSA-ES improves the objective value slowly. On the Rosenbrock and Rastrigin function, pcCMSA-ES fails to improve the objective value. The experimental results indicate that the population control of pcCMSA-ES works poorly as the noise is relatively strong.
4) *CMAES-APOP:* OPL-CMA-ES performs similar to CMAES-APOP on most test functions. OPL-CMA-ES converges faster than CMAES-APOP on the Rosenbrock and Rastrign function and achieves better final results in terms of the median value on the Tablet, Rastrigin, and Ackley function.

The experimental results show that OPL-CMA-ES generally performs significantly better than the compared algorithms on most of the test functions with relatively strong noise. It validates the effectiveness of the proposed mechanism of learning the population size.

V. EXPERIMENTS ON THE BBOB NOISY TESTBED

In this section, we comprehensively evaluate the performance of OPL-CMA-ES on the standard BBOB noisy testbed [35]. The BBOB noisy testbed consists of 30 noisy test functions and is commonly used as the standard benchmark for noisy optimization[2] [36]. We present a precise description of the benchmark functions of the BBOB noisy testbed in Table II. These test functions are classified into three subgroups according to the noise strength and the global structure of the landscape: 1) the subgroup with moderate noise; 2) the subgroup with severe noise; and 3) the subgroup with highly multimodal and severe noise.

The BBOB noisy testbed uses the expected runtime (ERT) and the empirical cumulative distribution function (ECDF) to evaluate algorithms' performance. The ERT estimates the ERT, i.e., the expected number of FEs to reach a given target function value [37], $f_{\text{target}} = f_{\text{opt}} + \Delta f$. It is summed over the number of FEs executed in all trials while the best function value did not reach f_{target} during and divided by the number of trials reached f_{target} [38]. The ECDF estimates the percentages of given target function values f_{target} reached for a given budget. The standard target function values f_{target} used in BBOB are $f_{\text{target}} = \min[f] + 10^k, k = \{-8, \ldots, 2\}$. The maximum FEs are set to $10^6 \times n$ for each objective function on BBOB noisy testbed. The initial mean $\mathbf{m}_0$ is randomly uniformly drawn from $[-4, 4]^n$, and $\sigma_0 = 4$.

In the experiments, we take some state-of-the-art noise-handling algorithms into the comparison. In addition to the compared algorithms APOP, pcCMSA, and UHCMA in Section IV,[3] we also include PSAaLmC [10], the MOS [28], and DE-PSO [27] in the comparison. These algorithms are of different categories of handling the noise. The MOS [39] is a hybrid algorithm that uses MOS structures to combine the advantages of different EAs. It achieves good performance on complicated multimodal functions and NOPs. The DE-PSO [27] is an improved PSO algorithm that applies the differential variation mechanism employed in DE to adapt

[1] The median run indicates the run with the median final solution value of each algorithm.

[2] Software and documentation are available from https://coco.gforge.inria.fr/.

[3] For convenience, we simplify OPL-CMA-ES, CMAES-APOP, pcCMSA-ES, and UH-CMA-ES to OPLCMA, APOP, pcCMSA, and UHCMA in this section.

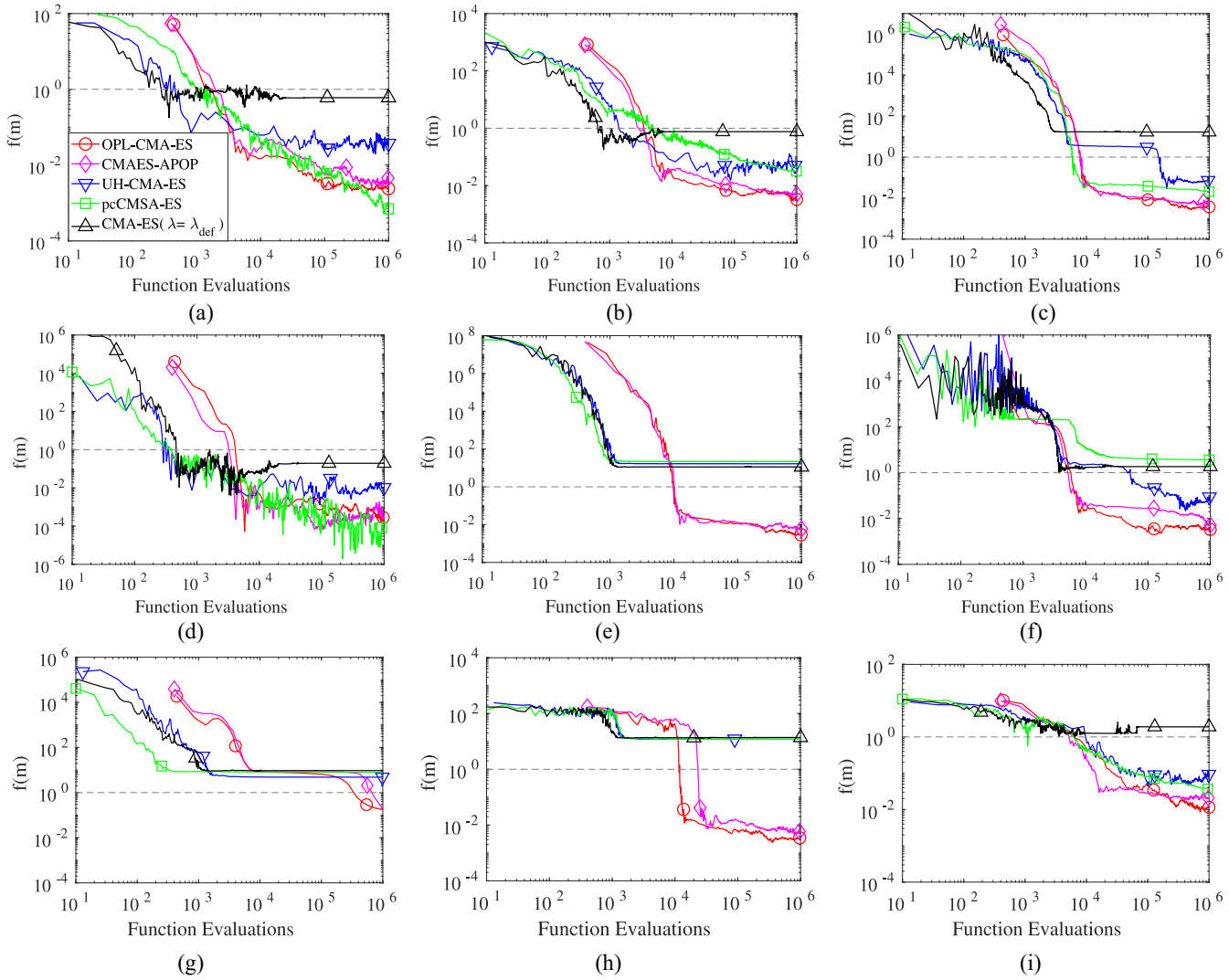

Fig. 2. Performance comparison on noisy functions with $n = 10$ and the noise strength $\sigma_\epsilon = 1$. Presented are objective trajectories of $f(\mathbf{m})$ of the median runs with the median final results out of 51 independent runs. (a) Sphere. (b) Benign ellipsoid. (c) Ellipsoid. (d) Diffpow. (e) Cigar. (f) Tablet. (g) Rosenbrock. (h) Rastrigin. (i) Ackley.

the velocity of particles and is robust against noise. The experimental data of the PSAaLMC, MOS, and DE-PSO are available at the BBOB workshops.[4]

A. Restart Strategy

CMA-ES can be considered as a robust local search method [40]. The restart strategy is commonly used to improve the robustness against multimodal or severe noisy problems. Hence, we restart the algorithm once any termination criteria of reaching the local minima are met [41].

In the detection of noise strength, the parameter θ represents the noise level that the algorithm can tolerate. With small θ, the algorithm can capture the subtle change in the rank and is more sensitive to the noise. Hence, we use different values of θ as well as the initial population size for restarts to detect the level of noise as follows.

First Run: In the first run, we expect that the algorithm can solve the unimodal function or the well-structured multimodal function with moderate noise. Hence, we use a relatively large threshold $\theta = 0.5$ and a small initial population size $\lambda_0 = \lambda_{\text{def}}$.

Second Run: After the termination of the first run, we investigate whether the objective function is a well-structured multimodal function with severe noise. To prevent the possible premature convergence, we use a large initial population size $\lambda_0 = k_n \times \lambda_{\text{def}}$ and moderate threshold $\theta = 0.3$.

Other Runs: If the first two runs fail to find the optimum, the objective function can be viewed as a weakly structured multimodal function with severe noise. We use a large initial population size $\lambda_0 = k_n \times \lambda_{\text{def}}$ and small threshold $\theta = 0.2$.

B. Experimental Results and Discussion

In this section, we present the experimental results of compared algorithms on the BBOB noisy testbed. More experimental results that include the ECDF data of each function and the numerical results of the ERT performance in all dimensions are presented in the supplementary material.

1) Expected Runtime Performance: Fig. 3 shows the ERT performance measured by the costed number of FEs of

[4]https://coco.gforge.inria.fr/doku.php?id=algorithms-bbob-noisy

TABLE II
MAIN FEATURES OF THE TEST FUNCTIONS OF BBOB NOISY TESTBED

Noise Strength	Noise Types			Test Functions
	Gaussian Noise	Uniform Noise	Cauchy Noise	
Moderate Noise	f_{101}	f_{102}	f_{103}	Sphere
	f_{104}	f_{105}	f_{106}	Rosenbrock
Severe Noise	f_{107}	f_{108}	f_{109}	Sphere
	f_{110}	f_{111}	f_{112}	Rosenbrock
	f_{113}	f_{114}	f_{115}	Step Ellipsoid
	f_{116}	f_{117}	f_{118}	Ellipsoid
	f_{119}	f_{120}	f_{121}	Different Powers
Highly Multi-modal with Severe Noise	f_{122}	f_{123}	f_{124}	Schaffer's F7
	f_{125}	f_{126}	f_{127}	Composite Griewank-Rosenbrock
	f_{128}	f_{129}	f_{130}	Gallagher's Gaussian Peaks 101-me

TABLE III
NUMBER OF SOLVED PROBLEMS FOR EACH ALGORITHM ON EACH DIMENSION

Dimension	2	3	5	10	20	40
OPLCMA	**30**	**30**	**29**	26	**24**	**22**
APOP	**30**	28	28	**28**	**24**	17
PSAaLmC	29	26	24	17	14	-
pcCMSA	21	15	11	6	5	-
UHCMA	19	19	14	11	9	6
MOS	**30**	27	25	19	16	10
DE-PSO	14	7	5	2	0	0

each algorithm to reach the target accuracy 10^{-8} on dimensions $2, 3, 5, 10, 20,$ and 40. The experimental results show that OPLCMA scales linearly on dimension in most of the test functions. OPLCMA generally outperforms PSAaLmC, pcCMSA, UHCMA, MOS, and DE-PSO in terms of the number of FEs to reach the required accuracy. It generally costs fewer FEs than APOP on $f_{101}, f_{102}, f_{103}$, and f_{106}, while APOP consumes fewer functions evaluations on $f_{105}, f_{116}, f_{119}$, and f_{124}. OPLCMA shows a remarkable improvement of the ERT on f_{126} in dimension 5, outperforming all the compared algorithms. The performance of OPLCMA differs from that of PSAaLmC, pcCMSA, and APOP, indicating the difference between the adaptation mechanisms. By adapting the population size, OPLCMA achieves robust performance on strong noise. Compared with UHCMA, OPLCMA achieves much more robust performance on strong noise and non-Gaussian noise, while UHCMA performs poorly on such problems. It indicates that adapting the number of evaluations can hardly work well on problems with strong noise and non-Gaussian noise.

Table III presents the number of solved problems among 30 test functions of each algorithm in the tested dimensions, where a problem is considered to be solved if the algorithm successfully reaches the required accuracy 10^{-8} within the limitation of FEs. OPLCMA can successfully solve more problems than PSAaLmC, pcCMSA, UHCMA, MOS, and DE-PSO on all the tested dimensions except that MOS solves the same number of functions on dimension 2. It should be noted that the performances of PSAaLmC, pcCMSA, UHCMA, MOS, and DE-PSO degenerate dramatically with the increasing of dimension, while OPLCMA achieves more robust performance. OPLCMA outperforms or performs comparatively to APOP on the tested dimensions. It should be noted that OPLCMA solves much more problems than all the compared algorithms on dimension 40, indicating that the proposed OPLCMA is more robust with increasing dimension.

2) ECDF Performance on Subgroups: Fig. 4 presents the ECDF performance of the compared algorithms for each subgroup. The test functions of each subgroup are characterized by different noise strengths and the global structure of the landscapes. The ECDF performance measures the fraction of functions of each subgroup that is solved at the different number of FEs. Generally, the algorithm that solves a larger fraction of functions with fewer FEs is the more favorable. Our main observations and discussions are as follows.

1) On the subgroup of functions f_{101}–f_{106} with moderate noise, OPLCMA successfully finds the solutions with the target accuracy on all dimensions as shown in the first column of Fig. 4. OPLCMA outperforms PSAaLmC, pcCMSA, UHCMA, and DE-PSO in all dimensions. With the same number of FEs, OPLCMA solves more functions. The final fractions of solved functions of PSAaLmC, pcCMSA, UHCMA, and MOS drop from 1.0 on dimension 2 to less than 0.7 on dimension 40. As the dimension increases from 2 to 40, their performances degenerate significantly and the final fractions of solved functions drop dramatically, while OPLCMA can solve all the functions successfully. Compared with APOP and MOS, OPLCMA generally runs faster as it costs fewer FEs.

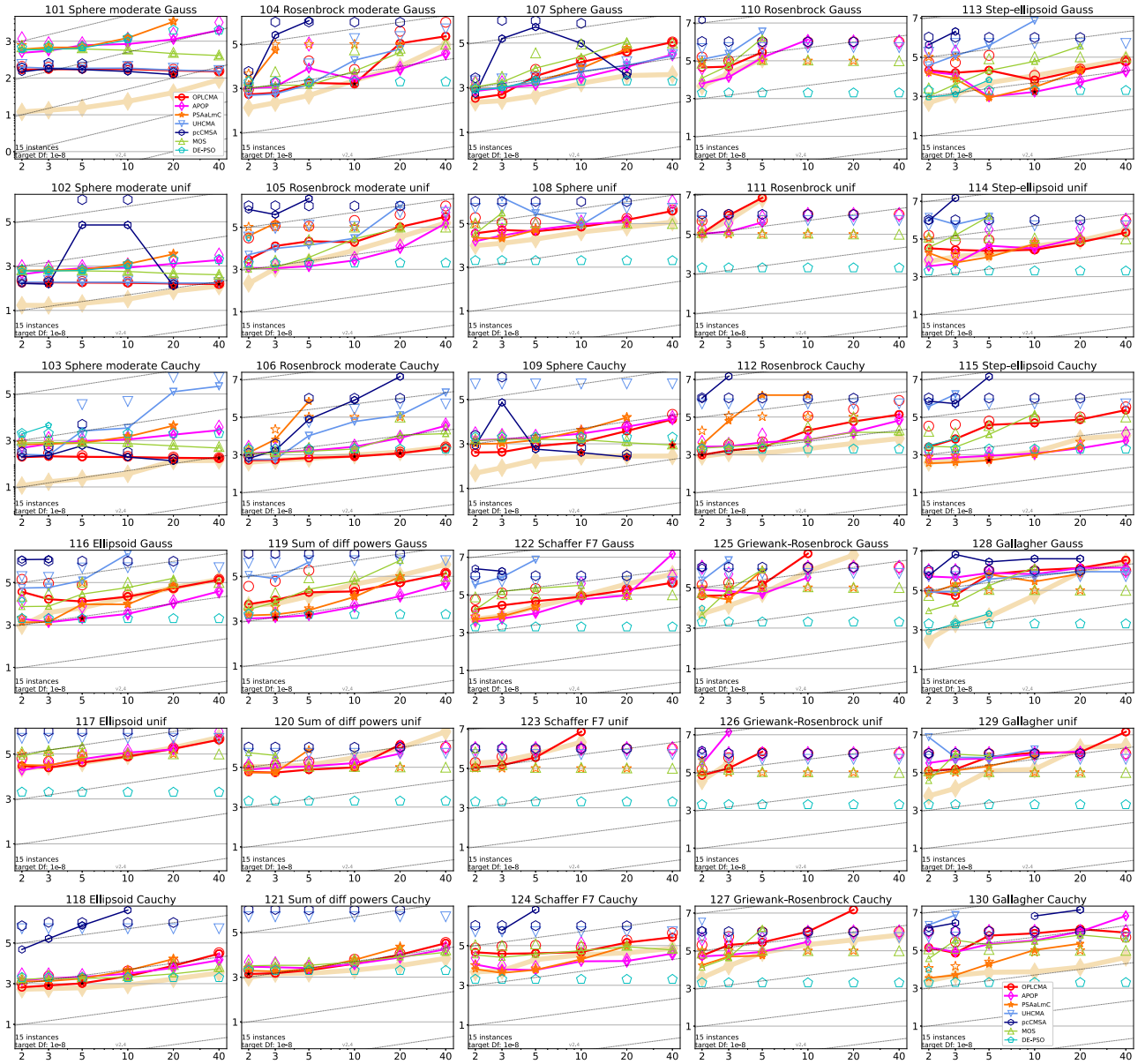

Fig. 3. ERT performance in number of FEs as $\log_{10}$ value divided by dimension for target function value 10^{-8} versus dimension. Slanted grid lines indicate quadratic scaling with the dimension. Light symbols give the maximum FEs from the longest trial divided by dimension. Black stars indicate a statistically better result compared to all other algorithms with $p < 0.01$ and Bonferroni correction number of dimensions (six). Legend: ○: OPLCMA, ◇: APOP, ⋆: PSAaLmC, ▽: UHCMA, ⬡: pcCMSA, △: MOS, and ⬠: DE-PSO.

2) On the subgroup of functions f_{107}–f_{121} with severe noise, OPLCMA performs competitively to APOP in low to moderate dimensions and outperforms APOP significantly in dimension 40. OPLCMA is superior to PSAaLmC, pcCMSA, UHCMA, MOS, and DE-PSO, especially as the dimension increases. The final fraction of solved functions by UHCMA drops from 0.84 on dimension 2 to less than 0.4 on dimension 40, indicating that UHCMA performs poorly on problems with severe noise. We observe the same performance drop on PSAaLmC, pcCMSA, MOS, and DE-PSO as the dimension increases.
3) On the subgroup of highly multimodal functions $f_{122} - f_{130}$ with severe noise, OPLCMA outperforms the compared algorithms in all dimensions, and the performance gaps enlarge with increasing dimension. APOP performs well in low to moderate dimensions, while it can hardly scale up to dimension 40. The performances of PSAaLmC, pcCMSA, UHCMA, MOS, and DE-PSO degenerate significantly as the dimension increases, as they can hardly solve the test functions with strong noise and weak global structure. Moreover, OPLCMA remarkably outperforms the best 2009 portfolio in 5-D and provides a significant improvement of the ECDF for f_{126} in 5-D (the detailed manner can be found in the supplementary material).

In summary, OPLCMA achieves superior performance than the compared algorithm on the BBOB noisy testbed, especially as the dimension increases. The final fraction of functions that OPLCMA can solve changes slightly, while that of the

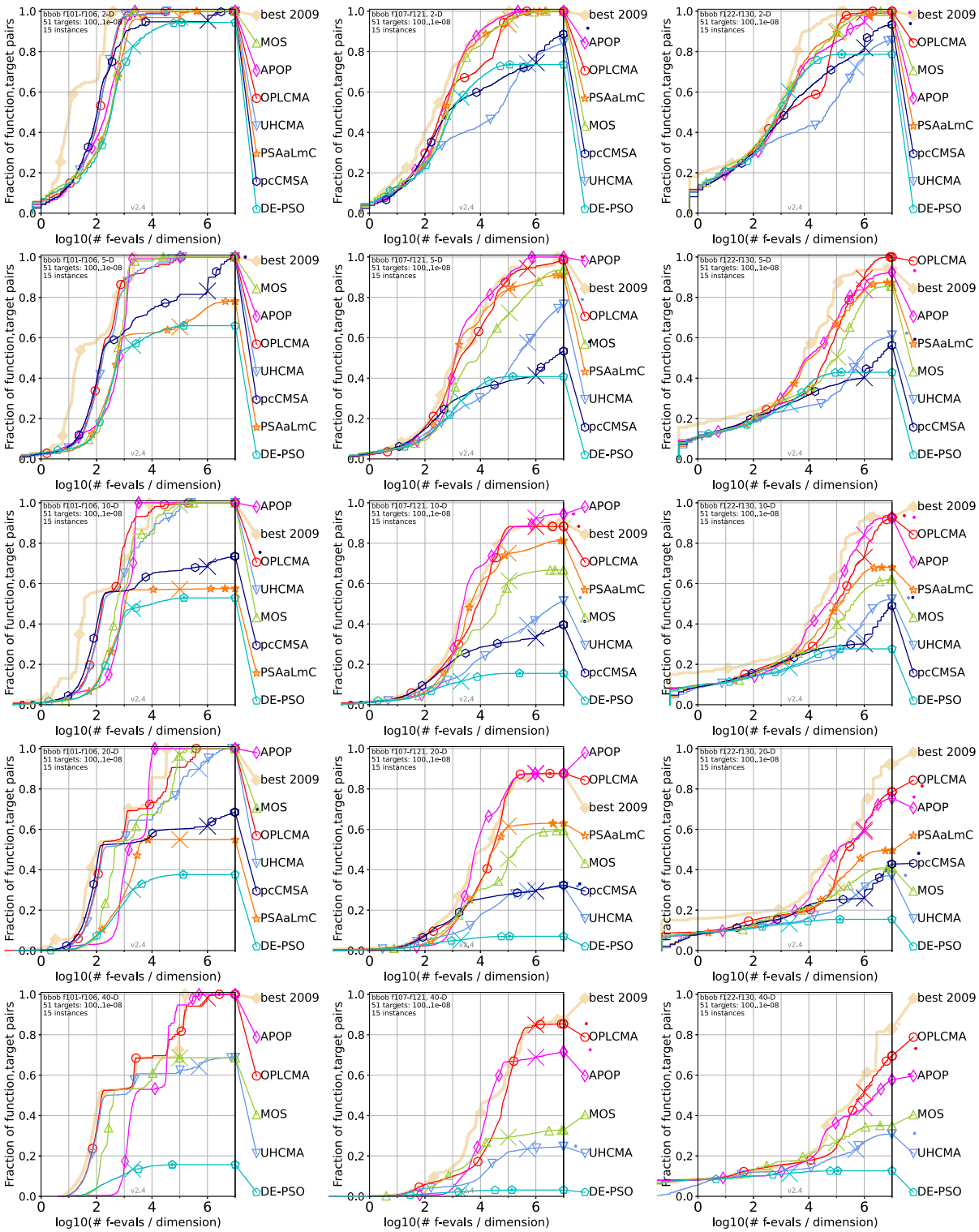

Fig. 4. ECDF performance for each subgroup in 2-, 5-, 10-, 20-, and 40-D. The light thick line with diamond markers shows the best algorithm from BBOB 2009 as a reference algorithm.

compared algorithms drops sharply. The experimental results validate the effectiveness of OPLCMA and the good scalability on dimension.

3) ECDF Performance on All Functions: Fig. 5 shows the ECDF performance for all functions in all dimensions. OPLCMA is superior to PSAaLmC, pcCMSA, UHCMA, MOS, and DE-PSO in all the compared dimensions. OPLCMA performs comparably to APOP in low to moderate dimensions and outperforms it significantly in dimension 40. OPLCMA can successfully solve more than 0.8 functions on dimension 40, while all compared algorithms are less than 0.7. The experimental results validate the effectiveness of the proposed OPL.

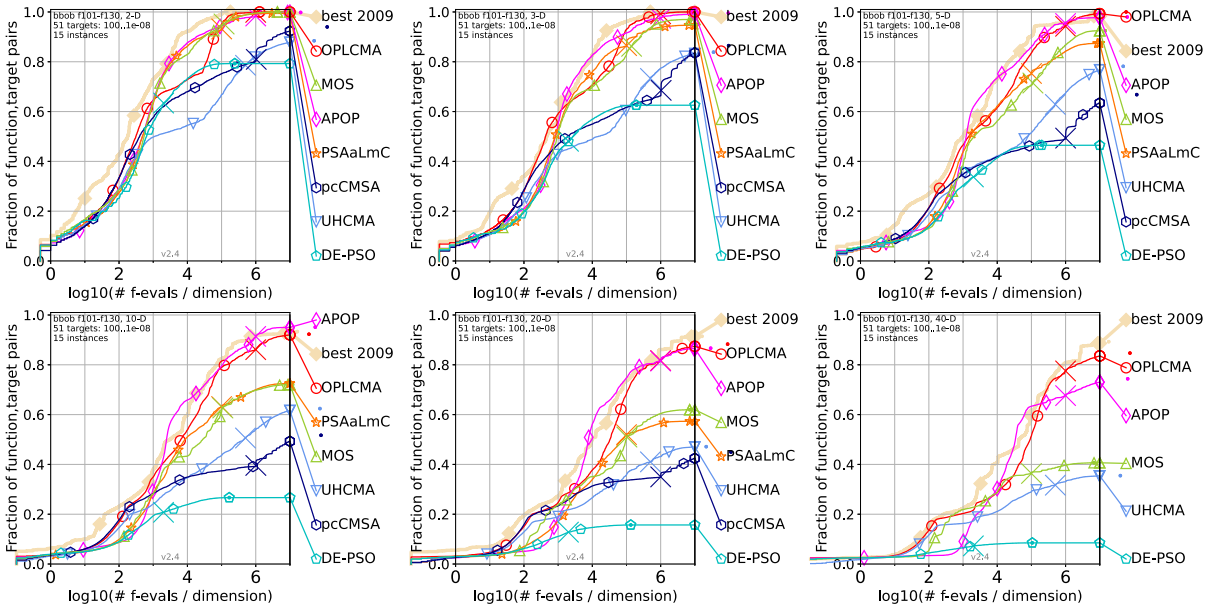

Fig. 5. ECDF performance for all functions in all tested dimensions. The light thick line with diamond markers shows the best algorithm from BBOB 2009 as a reference algorithm.

VI. Conclusion and Future Work

In this article, we have proposed a new method of OPL for handling NOPs and incorporated it with the CMA-ES denoted as OPL-CMA-ES. At each generation, we re-evaluate a fraction of the candidate solutions and measure the noise level by the rank changes of the re-evaluated candidate solutions, and then adapt the population size according to the noise level. The obtained OPL combines the advantages of both the explicit averaging and implicit averaging methods and overcomes their limitations. Furthermore, OPL does not depend on any specific evolutionary operations and adaptation mechanisms, and hence it represents a general framework for handling NOPs. It can be well suited for any ranking-based EAs without any modification of the algorithms.

We have conducted extensive experiments to evaluate the performance and behavior of OPL-CMA-ES. The experimental results show that OPL-CMA-ES is highly competitive on NOPs, especially when the noise level is strong. The experimental results validate the effectiveness of the proposed mechanism of OPL-CMA-ES. We further compared OPL-CMA-ES with some state-of-the-art variants on BBOB noisy testbed and showed the remarkable performance of OPL-CMA-ES.

In the future, we will investigate using OPL on the robust optimization with noise on the decision variable, and uncertainty in both the objective value and the decision variable. Furthermore, ESs have been successfully applied in solving deep reinforcement learning (DRL) problems [42]–[44]. The DRL problems are highly noisy in practice, yet the noise-handling techniques are rarely taken into account. The UH technique [45] and restart [46] are used to improve the performance of ESs. We will extend the proposed algorithm to large-scale variants [47], [48] and evaluate the performance on DRL problems in future work.

References

[1] D. Guo, Z. Nie, and L. Yan, "The application of noise-tolerant ZD design formula to robots' kinematic control via time-varying nonlinear equations solving," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 12, pp. 2188–2197, Dec. 2018.

[2] K. Esfandiari and M. Shakarami, "Bank of high-gain observers in output feedback control: Robustness analysis against measurement noise," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 4, pp. 2476–2487, Apr. 2021.

[3] P. Rakshit, A. Konar, and S. Das, "Noisy evolutionary optimization algorithms—A comprehensive survey," *Swarm Evol. Comput.*, vol. 33, pp. 18–45, Apr. 2017.

[4] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *IEEE Trans. Evol. Comput.*, vol. 9, no. 3, pp. 303–317, Jun. 2005.

[5] J. W. Kruisselbrink, "Evolution strategies for robust optimization," Ph.D. dissertation, Leiden Inst. Adv. Comput. Sci. (LIACS), Leiden Univ., Leiden, The Netherlands, 2012.

[6] P. Rakshit, A. Konar, S. Das, L. C. Jain, and A. K. Nagar, "Uncertainty management in differential evolution induced multiobjective optimization in presence of measurement noise," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 7, pp. 922–937, Jul. 2014.

[7] J. J. Merelo *et al.*, "There is noisy lunch: A study of noise in evolutionary optimization problems," in *Proc. 7th Int. Joint Conf. Comput. Intell. (IJCCI)*, vol. 1. Lisbon, Portugal, 2015, pp. 261–268.

[8] A. N. Aizawa and B. W. Wah, "Dynamic control of genetic algorithms in a noisy environment," in *Proc. 5th Int. Conf. Genet. Algorithms*, vol. 2, 1993, pp. 48–55.

[9] M. Hellwig and H.-G. Beyer, "Evolution under strong noise: A self-adaptive evolution strategy can reach the lower performance bound—The pcCMSA-ES," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 2016, pp. 26–36.

[10] K. Nishida and Y. Akimoto, "Population size adaptation for the CMA-ES based on the estimation accuracy of the natural gradient," in *Proc. Genet. Evol. Comput. Conf.*, 2016, pp. 237–244.

[11] O. Krause, "Large-scale noise-resilient evolution-strategies," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, 2019, pp. 682–690.

[12] S. Zhang, P. Chen, L. H. Lee, C. E. Peng, and C.-H. Chen, "Simulation optimization using the particle swarm optimization with optimal computing budget allocation," in *Proc. Winter Simul. Conf. (WSC)*, Phoenix, AZ, USA, 2011, pp. 4298–4309.

[13] J. Rada-Vilela, M. Zhang, and M. Johnston, "Optimal computing budget allocation in particle swarm optimization," in *Proc. 15th Annu. Conf. Genet. Evol. Comput.*, 2013, pp. 81–88.

[14] N. Hansen, A. S. P. Niederberger, L. Guzzella, and P. Koumoutsakos, "A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion," *IEEE Trans. Evol. Comput.*, vol. 13, no. 1, pp. 180–197, Feb. 2009.

[15] H.-G. Beyer and B. Sendhoff, "Evolutionary algorithms in the presence of noise: To sample or not to sample," in *Proc. IEEE Symp. Found. Comput. Intell.*, Honolulu, HI, USA, 2007, pp. 17–24.

[16] D. V. Arnold and H.-G. Beyer, "On the benefits of populations for noisy optimization," *Evol. Comput.*, vol. 11, no. 2, pp. 111–127, 2003.

[17] D. M. Nguyen and N. Hansen, "Benchmarking CMAES-APOP on the BBOB noiseless testbed," in *Proc. Genet. Evol. Comput. Conf. Companion*, 2017, pp. 1756–1763.

[18] Y. Ollivier, L. Arnold, A. Auger, and N. Hansen, "Information-geometric optimization algorithms: A unifying picture via invariance principles," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 564–628, 2017.

[19] K. Nishida and Y. Akimoto, "PSA-CMA-ES: CMA-ES with population size adaptation," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, 2018, pp. 865–872.

[20] S. Markon, D. V. Arnold, T. Back, T. Beielstein, and H.-G. Beyer, "Thresholding-a selection operator for noisy ES," in *Proc. Congr. Evol. Comput.*, vol. 1. Seoul, South Korea, 2001, pp. 465–472.

[21] J. Branke and C. Schmidt, "Selection in the presence of noise," in *Proc. Genet. Evol. Comput. Conf.*, 2003, pp. 766–777.

[22] F. Neri and A. Caponio, "A differential evolution for optimisation in noisy environment," *Int. J. Bio-Inspired Comput.*, vol. 2, nos. 3–4, pp. 152–168, 2010.

[23] E. Mininno and F. Neri, "A memetic differential evolution approach in noisy optimization," *Memetic Comput.*, vol. 2, no. 2, pp. 111–135, 2010.

[24] A. Ghosh, S. Das, B. K. Panigrahi, and A. K. Das, "A noise resilient differential evolution with improved parameter and strategy control," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Donostia, Spain, 2017, pp. 2590–2597.

[25] H.-G. Beyer and B. Sendhoff, "Robust optimization—A comprehensive survey," *Comput. Methods Appl. Mech. Eng.*, vol. 196, nos. 33–34, pp. 3190–3218, 2007.

[26] B. Liu, X. Zhang, and H. Ma, "Hybrid differential evolution for noisy optimization," in *Proc. IEEE Congr. Evol. Comput. (IEEE World Congr. Comput. Intell.)*, Hong Kong, 2008, pp. 587–592.

[27] J. García-Nieto, E. Alba, and J. Apolloni, "Particle swarm hybridized with differential evolution: Black box optimization benchmarking for noisy functions," in *Proc. 11th Annu. Conf. Companion Genet. Evol. Comput. Conf. Late Breaking Papers*, 2009, pp. 2343–2350.

[28] A. L. de la Fuente, "A framework for hybrid dynamic evolutionary algorithms: Multiple offspring sampling (MOS)," Ph.D. dissertation, Facultad de Informática, Universidad Politécnica de Madrid, Madrid, Spain, 2009.

[29] D. M. Nguyen, "Benchmarking a variant of the CMAES-APOP on the BBOB noiseless testbed," in *Proc. Genet. Evol. Comput. Conf. Companion*, 2018, pp. 1521–1528.

[30] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, "Natural evolution strategies," *J. Mach. Learn. Res.*, vol. 15, pp. 949–980, Mar. 2014.

[31] T. Suttorp, N. Hansen, and C. Igel, "Efficient covariance matrix update for variable metric evolution strategies," *Mach. Learn.*, vol. 75, pp. 167–197, May 2009.

[32] H.-G. Beyer and B. Sendhoff, "Simplify your covariance matrix adaptation evolution strategy," *IEEE Trans. Evol. Comput.*, vol. 21, no. 5, pp. 746–759, Oct. 2017.

[33] N. Hansen, "The CMA evolution strategy: A comparing review," in *Towards a New Evolutionary Computation*. Heidelberg, Germany: Springer, 2006, pp. 75–102.

[34] Y. Akimoto, A. Auger, and N. Hansen, "Quality gain analysis of the weighted recombination evolution strategy on general convex quadratic functions," *Theor. Comput. Sci.*, vol. 832, pp. 42–67, Sep. 2020.

[35] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noisy functions definitions," Inria, Le Chesnay-Rocquencourt, France, Rep. RR-6869, 2009.

[36] 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," *Optim. Methods Softw.*, vol. 36, no. 1, pp. 114–144, 2021.

[37] A. Auger and N. Hansen, "Performance evaluation of an advanced local search evolutionary algorithm," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2. Edinburgh, U.K., 2005, pp. 1777–1784.

[38] N. Hansen, A. Auger, S. Finck, and R. Ros, "Real-parameter black-box optimization benchmarking: Experimental setup," Université Paris Sud, Orsay, France, Institut National de Recherche en Informatique et en Automatique (INRIA) Futurs, Le Chesnay-Rocquencourt, France, Équipe TAO, Rep. 7215, 2012.

[39] A. LaTorre, S. Muelas, and J. M. Peña, "Benchmarking a MOS-based algorithm on the BBOB-2010 noisy function testbed," in *Proc. 12th Annu. Conf. Companion Genet. Evol. Comput.*, 2010, pp. 1725–1730.

[40] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2001.

[41] N. Hansen, "Benchmarking a BI-population CMA-ES on the BBOB-2009 noisy testbed," in *Proc. 11th Annu. Conf. Companion Genet. Evol. Comput. Conf. Late Breaking Papers*, 2009, pp. 2397–2402.

[42] T. Salimans, J. Ho, X. Chen, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," 2017, *arXiv:1703.03864*.

[43] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "Back to basics: Benchmarking canonical evolution strategies for playing atari," 2018, *arXiv:1802.08842*.

[44] L. Fuks, N. Awad, F. Hutter, and M. Lindauer, "An evolution strategy with progressive episode lengths for playing games," in *Proc. IJCAI*, 2019, pp. 1234–1240.

[45] N. Müller and T. Glasmachers, "Challenges in high-dimensional reinforcement learning with evolution strategies," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 2018, pp. 411–423.

[46] Z. Chen, Y. Zhou, X. He, and S. Jiang, "A restart-based rank-1 evolution strategy for reinforcement learning," in *Proc. IJCAI*, 2019, pp. 2130–2136.

[47] Z. Li and Q. Zhang, "A simple yet efficient evolution strategy for large scale black-box optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 5, pp. 637–646, Oct. 2018.

[48] Z. Li, Q. Zhang, X. Lin, and H.-L. Zhen, "Fast covariance matrix adaptation for large-scale black-box optimization," *IEEE Trans. Cybern.*, vol. 50, no. 5, pp. 2073–2083, May 2020.

Zhenhua Li (Member, IEEE) received the B.S. degree in mathematics from Zhengzhou University, Zhengzhou, China, in 2009, and the Ph.D. degree in computer science from the Department of Computer Science, City University of Hong Kong, Hong Kong, in 2018.

He is currently a Lecturer with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. His current research interests include evolutionary optimization and machine learning.

Shuo Zhang is currently pursuing the master's degree in computer science with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China.

His research interests include noisy optimization and robust optimization.

Xinye Cai (Member, IEEE) received the B.Sc. degree in information engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2004, the M.Sc. degree in electronic engineering from the University of York, York, U.K., in 2006, and the Ph.D. degree in electrical engineering from Kansas State University, Manhattan, KS, USA, in 2009.

He is an Associate Professor with the Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China, where he is currently leading an Intelligent Optimization Research Group, who has won the Evolutionary Many-Objective Optimization Competition at the Congress of Evolutionary Computation 2017. His current research interests include optimization, machine learning, and their applications.

Dr. Cai is an Associate Editor of *Swarm and Evolutionary Computation*.

Qingfu Zhang (Fellow, IEEE) received the B.S. degree in mathematics from Shanxi University, Taiyuan, China, in 1984, and the M.S. degree in applied mathematics and the Ph.D. degree in information engineering from Xidian University, Xi'an, China, in 1991 and 1994, respectively.

He is the Chair Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. His main research interests include evolutionary computation, optimization, neural network, data analysis, and their applications.

Dr. Zhang has been a Highly Cited Researcher in computer science since 2016. He is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and IEEE TRANSACTIONS ON CYBERNETICS. He is also an editorial board member of three other international journals.

Xiaomin Zhu (Member, IEEE) received the Ph.D. degree in computer science from Fudan University, Shanghai, China, in 2009.

He is currently an Associate Professor with the College of Information System and Management, National University of Defense Technology, Changsha, China. He has published more than 80 research articles in refereed journals and conference proceedings, such as IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON CLOUD COMPUTING, *Journal of Parallel and Distributed Computing*, IEEE CLOUD, and ICDCS. His research interests include scheduling and resource management in distributed systems.

Dr. Zhu serves on the Editorial Boards of *Future Generation Computer Systems* and *AIMS Big Data and Information Analytics*.

Zhun Fan (Senior Member, IEEE) received the B.S. and M.S. degrees in control engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1995 and 2000, respectively, and the Ph.D. degree in electrical and computer engineering from Michigan State University, Lansing, MI, USA, in 2004.

He is currently a Full Professor with Shantou University, Shantou, China. His major research interests include intelligent control and robotic systems, robot vision, and cognition.

Xiuyi Jia (Member, IEEE) received the Ph.D. degree in computer science from Nanjing University, Nanjing, China, in 2011.

He is currently an Associate Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing. His recent research focuses on machine learning, data mining, and computer vision.