Distributed Evolution Strategies with Multi-Level Learning for Large-Scale Black-Box Optimization

Qiqi Duan, Chang Shao, Guochen Zhou, Minghan Zhang, Qi Zhao, Yuhui Shi *Fellow, IEEE*

***Abstract*—In the post-Moore era, main performance gains of black-box optimizers are increasingly depending on parallelism, especially for large-scale optimization (LSO). Here we propose to parallelize the well-established covariance matrix adaptation evolution strategy (CMA-ES) and in particular its one latest LSO variant called limited-memory CMA-ES (LM-CMA). To achieve efficiency while approximating its powerful invariance property, we present a multilevel learning-based meta-framework for distributed LM-CMA. Owing to its hierarchically organized structure, Meta-ES is well-suited to implement our distributed meta-framework, wherein the outer-ES controls strategy parameters while all parallel inner-ESs run the serial LM-CMA with different settings. For the distribution mean update of the outer-ES, both the elitist and multi-recombination strategy are used in parallel to avoid stagnation and regression, respectively. To exploit spatiotemporal information, the global step-size adaptation combines Meta-ES with the parallel cumulative step-size adaptation. After each isolation time, our meta-framework employs both the structure and parameter learning strategy to combine aligned evolution paths for CMA reconstruction. Experiments on a set of large-scale benchmarking functions with memory-intensive evaluations, arguably reflecting many data-driven optimization problems, validate the benefits (e.g., effectiveness w.r.t. solution quality, and adaptability w.r.t. second-order learning) and costs of our meta-framework.**

***Index Terms*—Black-box optimization (BBO), distributed optimization, evolution strategies (ESs), large-scale optimization (LSO), parallelism.**

I. Introduction

As both Moore's law [1], [2] and Dennard's law [3], [4] come to end [5], main gains in computing performance will come increasingly from the top of the computing stack (i.e., algorithm developing, software engineering, and hardware streamlining) rather than the bottom (semiconductor technology) [6]. Refer to e.g., the latest *Science* review [6] or the *Turing* lecture [7] for an introduction to the modern computing stack. As recently emphasized by two Turing-Award winners (i.e., Hennessy and Patterson), *"multicore [8] shifted responsibility for identifying parallelism and deciding how to exploit it to the programmer..."* [7]. To follow this multi/many-core trend, in this paper we explore the parallelism of evolutionary algorithms (EAs [9]), since intuitively their population-based (random) sampling strategies [10], [11] are well-suited for massive parallelism [12]–[15].

Specifically, we consider the derandomized evolution strategy with covariance matrix adaptation (CMA-ES [16]–[18]) and in particular its one latest variant called limited-memory CMA (LM-CMA [19], [20]) for large-scale black-box optimization (BBO). As stated in the popular *Nature* review [12], *"CMA-ES is widely regarded as (one of) the state of the art in numerical (black-box) optimization"* with competitive performance on many benchmarking functions [21]–[25] and challenging applications (such as [26]–[31], just to name a few). Although typically its absolute runtime is of minor relevance [17] for low-dimensional (e.g., ≤ 50) cases, it cannot be ignored in the distributed computing context for large-scale (e.g., ≥ 1000) optimization (LSO) because of its (at least) quadratic complexity w.r.t. each sampling. Since its limited-memory variant (LM-CMA) can fit more for memory hierarchy and distributed communication, our aim is to extend LM-CMA to the modern cloud/clustering computing environment for LSO, in order to show effectiveness [32] while approximating the attractive *invariance* property of CMA-ES [33] as much as possible.

In their seminal paper [17], Hansen and Ostermeier pointed out four fundamental demands for ESs: adaptation, performance, invariance [34], and stationarity (i.e., unbiasedness under random selection). Clearly, these demands[1] are also highly desirable for any distributed ES (DES), in order to obtain efficiency and generalization/transferability [36]. To meet these demands in the distributed computing environment, we adopt the multilevel learning perspective for evolution (recently published in *PNAS* [37], [38]) to model and design the efficient DES framework, wherein Meta-ES [39], [40] and LM-CMA could be naturally combined together to enjoy the best of both worlds. Three main contributions of our paper to DES for large-scale BBO are presented as below:

1) We first analyze the parallelism opportunities and challenges of two different ES families (i.e., CMA-ES and Meta-ES) under two common models (i.e., master-salve and island). See Section II for details. For CMA-ES, efficient (distributed shared) memory usage is a prerequisite to design its distributed algorithm, and instead its large-scale variants are preferred as the baseline computing unit because of their much lower (e.g., $O(n \log n)$) complexity. For Meta-ES, the choice of its inner-

This work is supported by the Shenzhen Fundamental Research Program under Grant No. JCYJ20200109141235597 and the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant No. 2017ZT07X386.

Qiqi Duan is with Harbin Institute of Technology, Harbin, China and Southern University of Science and Technology, Shenzhen, China. (e-mail: 11749325@mail.sustech.edu.cn)

Chang Shao is with University of Technology Sydney, Sydney, Australia.

Minghan Zhang is with University of Warwick, Coventry, UK.

Guochen Zhou, Qi Zhao, and Yuhui Shi are with Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. (e-mail: shiyh@sustech.edu.cn).

Manuscript received X X, 2023.

[1] As previously stated in the classical ES review by Beyer and Schwefel [35], if some principle is deviated when we design the ES-based variant, the designed optimizer often needs to be well-tested.

ES loop significantly impacts its convergence rate, since the fixed global step-size adaptation used in its inner-ESs may be not optimal for many challenging problems when a long isolation time is used for population diversity as well as low communication costs. We argue that multilevel learning for biological evolution (MLE) is a natural way to hierarchically combine LM-CMA with Meta-ES under distributed computing (see Section III for details).

2) Inspired by the recent evolution theory (i.e., MLE) [37], we propose a multilevel learning-based meta-framework for DES to exploit *spatio-temporal* information (if available) on-the-fly to accelerate global convergence while maintaining meta-population diversity under the affordable cloud/clustering computing platform. Within this meta-framework, four following key design choices for DES are made reasonably, in order to balance search efficiency (w.r.t. convergence rate, absolute runtime, and diversity) and extra computing cost brought by distributed enhancements (e.g., distributed scheduling, load balancing, fault-tolerance on computing nodes, and data exchanges over the network, etc.) [41].

First, owing to its hierarchically organized structure, Meta-ES is well-suited to implement the multilevel meta-framework for DES. Generally, the (unique) outer-ES controls only strategy parameters, which parameterize the (typically Gaussian) search/mutation distribution [42], while parallel inner-ESs evolve independent sub-populations (with different strategy parameters initialized via the outer-ES level) within each isolation time. Unlike all of previous Meta-ESs which employ *fixed* strategy parameters for inner-ESs, here we use the computationally efficient LM-CMA as the base to learn the (local) topology, some of which are then collected to approximate the global topology at the outer-ES level. Refer to Section III-A for details.

Second, at the outer-ES level, the mean update of search distribution may suffer from stagnation when the *elitist* strategy from previous Meta-ESs is used (to our knowledge, this finding is dated back to Schwefel [43]). If only the *multi-recombination* strategy is used [44], it may suffer from the regression issue on some functions (e.g., Cigar). In our meta-framework, both the elitist and multi-recombination strategy are used in a parallel fashion, to enjoy the better of two worlds (see Section III-B for details).

Third, the global step-size (also called mutation strength) self-adaptation for DES combines the less-known Meta-ES strategy with the well-known cumulative step-size adaptation (CSA), in order to exploit both the *spatial* and *temporal* (non-local) information. As stated previously by Rudolph [45], the optimization/control problem of strategy parameters is multi-modal [17] and noisy (from sampling). Although they sometimes can be seen as hyper-parameters when an offline perspective is used [46], [47], strategy parameters themselves are usually controlled in an online fashion during evolution, in order to timely utilize (possibly) topology information to maximize convergence progress (see Section III-C for details).

Fourth, to keep a sensible trade-off between efficiency and stability for DES, we extend the collective learning strategy (originally proposed by Schwefel [48]) to the distributed computing scenario mainly from the following two aspects: i) after each isolation time all evolution paths from parallel LM-CMAs will be aggregated (via weighted averaging at the outer-ES level) as the shared baseline to reconstruct initial covariance matrix for inner-ESs (i.e., LM-CMAs) of the next isolation time; ii) different LM-CMA structures, simply parameterized as the total number of evolution paths for CMA reconstruction, are adapted at the outer-ES level, in order to approximate the global geometry of the fitness landscape as soon as possible (see Section III-D for details).

3) To validate the effectiveness [49] of our proposed meta-framework, we conduct numerical experiments on a large set of large-scale (i.e., 2000-d) BBO functions. The standard rotation operation used to avoid separability results in memory-expensive fitness evaluations, arguably reflecting the feature of many real-world data-driven optimization problems. Experimental results show the benefits (and also cost) of our proposed DES meta-framework for large-scale BBO (see Section IV for details). In principle, this meta-framework can also be extended to other evolutionary algorithms with modifications.

II. Related Works

In this section, we review only parallel/distributed versions and large-scale variants of ES, since there have been some well-written reviews for ES (e.g., [21], [35], [50]–[53]) up to now. We also analyze the possible opportunities and challenges of two different ES families (i.e., CMA-ES and Meta-ES) under two common parallelism models (i.e., master-salve and island).

A. Parallel/Distributed Evolution Strategies

Recent advances in parallel/distributed computing, particularly cloud computing [54]–[56], provide new advantages and challenges for evolutionary algorithms (EAs) [12], [57]. Although it comes as no surprise that parallelism is not a panacea for all cases [58]–[60] distributed evolutionary strategies (DES) are playing an increasingly important role in large-scale BBO in the post-Moore era [6], [61]. Refer to e.g., the recently proposed *hardware lottery* [62] for interesting discussions. Note that here we focus on model-level or application-level parallelism[2] (rather than instruction-level parallelism [7]).

According to [63], one of the first to parallelize ESs is [64], where an *outdated* vector computer was employed in 1983. In the early days of ES research, Schwefel [43] used the classical master-slave [65]–[67] model to conduct evolutionary (collective) learning of variable scalings on parallel architectures[3]. However, only a simulated (not realistic) parallel environment was used in his experiments, where the costs of data communication, task scheduling, and distributed

[2] Although some researchers viewed some distributed EAs as a new class of meta-heuristics, in this paper we adopt a conservative perspective, that is, distributed EAs are seen as a performance enhancement under distributed computing.

[3] The master-slave model (aka farmer/worker [68], [69]) is typically used for computationally-intensive fitness evaluations such as optimization of aircraft side rudder and racing car rear wing [68]), etc. The well-established Amdahl's law [70], [71] can be used as an often useful speedup estimation.

fault-tolerance were totally ignored. It may over-estimate the convergence performance of DES. This issue existed mainly in early DES studies such as [72]–[74] given the fact that at that time commercial parallel/distributed computers were not widely available and Moore's law worked well.

Rudolph [73] used the popular island (aka coarse-grained [75]) model for DES. However, it only considered the simple migration operation and did not cover the distributed self-adaptation of individual step-sizes, which often results in relatively slow convergence [76]. Although Neumann et al. [77] provided a theoretical analysis for the migration setting, its discrete assumptions and artificially constructed functions cannot be naturally extended to continuous optimization. Overall, there have been relatively rich theoretical works (e.g., [78]–[81]) on parallel algorithms for discrete optimization while there is little theoretical work (e.g., [82]) on parallel EAs for continuous optimization, up to now.

Wilson et al. [83] proposed an asynchronous communication protocol to parallelize the powerful CMA-ES on cloud computing platforms. When the original CMA-ES was used as the basic computing unit for each CPU core, however, under its quadratic computational complexity the problem dimensions to be optimized are often much low (e.g., only 50 in their paper) by the modern standard for large-scale BBO. Similar issues are also found in existing libraries such as pCMALib [84], Playdoh [85], OpenFPM [86] and [87]. Glasmachers [88] updated strategy parameters asynchronously for Natural ES (NES) [89], a more principal version for ES. The runtime speedup ratio obtained was below 60% on the 8-d Rosenbrock function, which means the need of improvements. Reverse and Jaeger [90] designed a parallel island-based ES called piES to optimize a non-linear (62-d) systems biology model [91]. In piES, only individual step-sizes were self-adapted for each island (the CMA mechanism was ignored). The speedup formulation used in their paper considered only the runtime but not the solution quality, which may lead to over-optimistic conclusions in many cases.

Recently, Cuccu et al. [92] proposed a DES framework called DiBB based on the partially separable assumption [93], [94]. Although it obtained significant speedups when this assumption is satisfied, DiBB did not show obvious advantages than the serial CMA-ES on ill-conditioned non-separable landscapes [92]. Kucharavy et al. [95] designed a Byzantine-resilient [96] consensus strategy for DES. However, they did not explicitly consider the improvement of search performance and no performance data was reported in their paper. Rehbach et al. [97] used the classical (1+1)-ES as the local searcher for parallel model-based optimization on a small-sized (i.e., 16-core) parallel hardware. To our knowledge, they did not consider the more challenging distributed computing scenarios.

Another (perhaps less-known) research line for DES is Meta-ES (also referred to as Nested-ES, first proposed by Rechenberg [98], [99]) which organizes multiple independent (parallel) ESs hierarchically [39], [100]. Although there have been relatively rich theoretical works on different landscapes (e.g., parabolic ridge [101], sphere [102], noisy sphere [40], sharp ridge [103], ellipsoid [104], conic constraining [105]), to our knowledge all these theoretical models do not take overheads from distributed computing [49] into account. Furthermore, although these works provide valuable theoretical insights to understand Meta-ES, all the inner-ESs used in their models are relatively simple from a practical viewpoint.

In this subsection, we omit the diffusion (also called neighborhood or cellular [106], [107]) model [108], as it rarely used in the distributed computing scenario considered in our paper. For a more general introduction to distributed EAs, refer to e.g., two recent survey papers [109], [110].

B. Large-Scale Variants of CMA-ES

In this subsection, we review large-scale variants of CMA-ES through the lens of distributed computing. For a detailed introduction, refer to e.g., [24], [111]–[113] and references therein.

Because the standard CMA-ES has a quadratic time-space complexity, it is difficult to directly distribute it on cloud or clustering computing platforms for large-scale BBO. A key point to alleviate this issue is to reduce the computational complexity of the CMA mechanism, in order to fit better for the (distributed) memory hierarchy. Till now, different ways have been proposed to improve its computational efficiency: 1) exploiting the low-rank structure, e.g., [112]–[116]; 2) making the separable assumption, e.g., [117], [118]; 3) inspiring from L-BFGS, e.g. [19], [20], [119]; 4) seeking computationally more efficient implementations [120]–[126].

For many large-scale variants of CMA-ES, one obvious advantage against its standard version is their much lower time-space complexity (e.g., $O(n \log n)$ for LM-CMA). To the best of our knowledge, however, their distributed extensions are still rare up to now, despite of their clear advantages on large-scale BBO.

One key challenge for DES lies in the trade-off between (computation) simplicity and (model) flexibility. On the one hand, we need to keep the structure of CMA, which can be simply parameterized as the number of evolution paths to reconstruct, as simple as possible, in order to fit better for memory hierarchy and reduce communication costs. On the another hand, we also expect to maintain the well-established invariance property as much as possible, in order to keep the flexible expressiveness/richness of model (resembling the second-order optimization method) [127]. To achieve such a trade-off, an efficient adaptive strategy (particularly at the meta-level) is highly desirable, which is the goal of our paper.

III. A Multilevel Meta-Framework for DES

In this paper we propose a multilevel-based meta-framework for distributed evolution strategies (DES) to minimize[4] the large-scale BBO problem $f(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$, where $n \gg 100$ is the dimensionality. Very recently, a research team from the biologist Koonin has presented a mathematical model of evolution through the lens of multilevel learning in their two *PNAS* papers [37], [38]. Inspired by this evolution theory,

[4]Without loss of generality, the maximization problem can be easily transferred to the minimization problem by simply negating it.

our meta-framework for DES mainly involves hierarchical organization of distributed computing units (via Meta-ES), multilevel selection[5], and collective learning of parameterized search/mutation distributions on structured populations [132]. While this new evolution theory is interpreted in a mathematical way, our meta-framework needs to be well-interpreted from an *optimizer* view of point, as shown below in detail.

A. Hierarchical Organization of LM-CMA via Meta-ES

As pointed out by Rudolph, the design of DES should be well-aligned to the target hardware (see [133] for classification), in this paper we consider the clustering/cloud computing platform consisting of a number of independent high-performing Linux servers, each of which owns one shared memory and multiple CPU cores. These Linux servers are connected via a high-speed local area network (LAN).

The population structure plays a fundamental role on the search dynamics of DES [134]. To obtain a statistically stable learning process, we use the hierarchically organized structure from Meta-ES to control/evolve parallel LM-CMA. In principle, other LSO variants of CMA-ES could also be used here as the basic computing unit on each CPU core. When many distributed computing units are available, large populations are highly desirable for many cases such as multimodality and noisiness [135]–[137]. Simply speaking, Meta-ES is an efficient way to build a structured level for the large population, in order to reduce communication costs.

For Meta-ES, one key hyper-parameter is the isolation time τ, which controls the communication frequency at different levels (i.e., between the outer-ES and all parallel inner-ESs). It is no surprise that the optimal setting of isolation time τ is problem-dependent. Generally, the longer the isolation time, the more diverse (slower) the population (local convergence); and vice versa. Furthermore, the longer the isolation time, the lower (slower) the communication cost (learning progress); and vice versa. To attain a satisfactory performance for DES, our meta-framework needs to keep reasonable trade-offs between population diversity and convergence rate, and between communication cost and learning progress, which will be tackled in the following subsections.

B. Distribution Mean Update at the Outer-ES Level

At the outer-ES level, the *elitist* [138]–[140] or *weighted multirecombination* strategy is used to initialize the distribution mean of each inner-ES from the next isolation time, according to a controllable ratio μ' (e.g., 1/5 vs 4/5)[6]. The rationale behind this parallel update strategy is presented in the following: If only the *elitist* strategy is used, the parallel search process may suffer from stagnation; if only the *multirecombination* strategy is used, the parallel search process may suffer from the regression issue on some functions (e.g., with a predominated search direction). Note that for simplicity, the default distribution mean update is used for each inner-ES as the same as LM-CMA within each isolation time τ.

In the outer-ES, the *weighted multirecombination* update of its distribution mean m' after each isolation time is mathematically calculated as

$$\mathbf{m}' = \sum_{i=1}^{\mu'} w'_{i;\lambda'} \mathbf{m}_{i;\lambda'}, where \sum_{i=1}^{\mu'} w'_{i;\lambda'} = 1, \qquad (1)$$

where μ' is the used (selected) number of all (λ') parallel inner-ESs, $w'_{i;\lambda'}$ and $\mathbf{m}_{i;\lambda'}$ are the weight and distribution mean of the ith-ranked[7] inner-ES, respectively. Even at the outer-ES level, we still follow the standard practice (that is, the higher the ranking, the larger the weight) to set all the weights. Refer to e.g., Arnold's theoretical analysis [44] for a better understanding.

C. Spatiotemporal Global Step-Size Adaptation (STA)

As previously pointed out by Rudolph [45], the (online) control problem of strategy parameters is multi-modal and noisy. Even if the optimization problem is deterministic rather than noisy, the random sampling nature from the inner-ES level makes it difficult to adapt the global step-size at the outer-ES level. To obtain a reliable estimation, a *relatively long* isolation time τ may be preferred, which also brings some extra benefits w.r.t. communication costs and fault tolerance.

In order to exploit both the *spatio* and *temporal* (non-local) information on-the-fly, our meta-framework combines the less-known Meta-ES strategy with the well-known CSA strategy (or its recent population-based or success-based variants for LSO)[8]. Specifically, three parallel design strategies are used at the outer-ES to balance adaptation speed and meta-population diversity [141]: 1) when the elitist strategy is used for the distribution mean update, the same *elitist* strategy is also used to update the global step-size of the corresponding inner-ES to obtain the relatively stable evolution process; 2) given a predefined proportion (e.g., 1/5), some inner-ESs *mutate* the weighted multi-recombined global step-size σ' as $\sigma_i \sim \mathcal{U}(\sigma' * a, \sigma' * b)$, where $0 < a < 1 < b$, implicitly based on the *strong causality* assumption [9]; 3) otherwise, the global step-size will be *uniformly* sampling from a reasonable search range[10], in order to maintain diversity and reduce the risk of getting trapped into local minima.

The weighted multirecombination for the global step-size σ' of the outer-ES should be done (somewhat) in an *unbiased* way:

[5]For the modern theory regarding to evolutionary transitions in individuality [128], [129], multilevel selection is regarded as a crucial factor to understand life's complexification [130], [131].

[6]For simplicity, this hyper-parameter μ' is also used for the weighted multirecombination strategy of the outer-ES. See (1) for details.

[7]For minimization, the lower the fitness (cost), the higher the ranking.

[8]In this paper, we do not modify the CSA-style strategy for all inner-ESs. Instead, we focus on the global step-size adaptation at the outer-ES level, which is crucial for DES.

[9]For simplicity, in this paper we follow the common suggestion from Meta-ES and set $a = 0.3$ and $b = 3.3$, respectively (note that $1/3.3 \approx 0.3$ leads to unbiasedness in the logarithmic scale).

[10]In practice, a reasonable search range seems to be easier to set than a reasonable value.

$$\sigma' = \sum_{i=1}^{\mu'} \frac{w'_{i;\lambda'}\sigma_{i;\lambda'}}{\sqrt{\sum_{i=1}^{u'} w'_{i;\lambda'}}}, \tag{2}$$

where the denominator $\sqrt{\sum_{i=1}^{u'} w'_{i;\lambda'}}$ ensures $\sigma' \sim \mathcal{N}(0,1)$ at the logarithmic scale under neutral selection (one of basic design principles from the ES community). Different from the CSA, STA does not use the exponential smoothing method at the outer-ES level, since the temporal information has been well exploited by each inner-ES and it is hard to set the corresponding learning rate and decaying factor (undoubtedly, it is expensive to set them at the outer-ES level).

D. Collective Learning of CMA on Structured Populations

The most prominent feature of CMA-ES appears to be its adaptive encoding (i.e., invariance against affine transformation) ability, especially on non-separable, ill-conditioned problems. As a general-purpose black-box optimizer, we expect DES to keep this powerful feature as much as possible. In order to be communication-efficient, however, we need to properly compress the standard $n \times n$ covariance matrix to fit for the distributed shared memory, but which may destroy the highly desirable invariance property. In this paper, we choose to use one of its large-scale variants (i.e., LM-CMA) as the basic computing unit on each CPU core, in order to reduce the communication cost after each isolation time τ.

The simplified form of CMA, derived by Beyer and Sendhoff [121], is presented as

$$\begin{aligned}\mathbf{C}^{t+1} \leftarrow & \mathbf{C}^{t}\Big\{\mathbf{I} + \frac{c_1}{2}(\mathbf{p}^{t+1}(\mathbf{p}^{t+1})^T - \mathbf{I}) \\ & + \frac{c_\mu}{2}(\sum_{i=1}^{\mu} w_i \mathbf{z}_{i;\lambda}^t (\mathbf{z}_{i;\lambda}^t)^T - \mathbf{I})\Big\},\end{aligned} \tag{3}$$

where $\mathbf{C}^t$ is the transformation matrix at the t-th generation (another form of the covariance matrix to avoid eigen-decomposition), $\mathbf{I}$ is the identity matrix, $\mathbf{p}^{t+1}$ is the evolution path at the $(t+1)$-th iteration, w_i is the weight for the i-th ranked individual, $\mathbf{z}_{i;\lambda}^t$ is the realized random sample from the standard normal distribution for the i-th ranked individual, μ is the number of parents of the inner-ES, c_1 is the coefficient of the rank-one update [17], and c_μ is the coefficient of the rank-μ update [74], respectively.

After omitting the update-μ update, the sampling procedure can be significantly reduced to

$$\begin{aligned} d_i^t = & \Big((1-\frac{c_1}{2})\mathbf{I} + \frac{c_1}{2}\mathbf{p}^1(\mathbf{p}^1)^T\Big) \\ & \times \Big((1-\frac{c_1}{2})\mathbf{I} + \frac{c_1}{2}\mathbf{p}^2(\mathbf{p}^2)^T\Big) \\ & \times \cdots \\ & \times \Big((1-\frac{c_1}{2})\mathbf{I} + \frac{c_1}{2}\mathbf{p}^{t-1}(\mathbf{p}^{t-1})^T\Big) \\ & \times \Big((1-\frac{c_1}{2})\mathbf{I} + \frac{c_1}{2}\mathbf{p}^{t}(\mathbf{p}^{t})^T\Big)\mathbf{z}_i^t. \end{aligned} \tag{4}$$

It is worthwhile noting that the above equation should be calculated from right to left, in order to get a linear complexity for each operation. Owing to the limit of pages, please refer to [20] for detailed mathematical derivations. To reduce the overall computational complexity, only a small amount of evolution paths (parameterized as n^e here) are used in all limited-memory LSO variants but with different selection rules. Because the successive evolution paths usually exhibit relatively high correlations, a key point is to make a diverse baseline of evolution paths for the covariance matrix reconstruction. In practice, different problems often have different topology structures and need different fitting structures, which naturally lead to the structure learning problem.

For properly approximating CMA under distributed computing, our meta-framework employs two adaptive distributed strategies for structure and distribution learning, respectively. First, as the structure learning often operates at a relatively slow-changing scale and controls the richness of distribution model, we implicitly adapt the total number of reconstructed evolution paths n^e via the *elitist* strategy, in order to obtain a reliable learning progress at the outer-ES level. In other worlds, we save a certain elitist ratio[11] as some (but not all) parallel inner-ESs for the next isolation time and for the remaining parallel inner-ESs we sample n^e *uniformly* in a reasonable setting range[12], in order to maintain the diversity of structure learning at the outer-ES level. Note that for each inner-ES, its reconstructed structure is always fixed within each isolation time.

For collective learning of search distributions under distributed structured populations, our meta-framework uses a simple yet efficient *weighted multirecombination* strategy to combine the evolution paths from the elitist inner-ESs into a shared pool of reconstructed evolution paths $\mathbf{P}'$ for the next isolation time. Cautiously, owing to possibly heterogeneous shapes from structure learning, we need to align the weighted multirecombination operation as follows (first $\mathbf{p}'$ is initialized as a $\max\limits_{i=1,\ldots,\mu'}(n_i^e) \times n$ zero matrix after each isolation time):

$$\mathbf{P}'[-n_i^e:] \mathrel{+}= \frac{w'_{i;\lambda'}}{\sqrt{\sum_{i=1}^{u'} w'_{i;\lambda'}}}\mathbf{P}_i[-n_i^e:](i=1,\cdots,\mu'), \tag{5}$$

where $[-n_i^e:]$ denotes all indexes starting from the last n_i^e column to the end, $\sqrt{\sum_{i=1}^{u'} w'_{i;\lambda'}}$ ensures unbiasedness under neutral selection, and $\mathbf{p}_i$ is a pool of reconstructed evolution paths from the i-th ranked inner-ES, respectively.

E. A Meta-Framework for DES

Here we combine all the aforementioned design choices into our distributed meta-framework, as presented in **Algorithm 1**[13]. To implement our meta-framework, we select one state-of-the-art clustering computing software called *ray* [143] as the key engine of distributed computing. As compared with other

[11] It is set to μ' for consistency and simplicity in this paper.

[12] In practice, the setting of the search range of n^e depends on the available memory in the used distributed computing platform, which is easy to obtain.

[13] Rinnooy Kan and Timmer [142] from the mathematical programming community considered a multi-level method for stochastic optimization in the context of *single linkage*. However, our approach is orthogonal to their method.

Algorithm 1 A Multilevel-based Meta-Framework for DES.

Input: λ': the number of all parallel inner-ESs (LM-CMAs)
μ': the number of elitists for the outer-ES
$\mathbf{m}_i$: the distribution mean of the i-th inner-ES
σ_i: the global step-size of the i-th inner-ES
$\mathbf{P}_i$: a pool of evolution paths of the i-th inner-ES
τ: the isolation time (i.e., runtime of each LM-CMA)
σ': the global step-size of the outer-ES
$\sigma_{\max}$: maximally possible value of global step-size
n_i^e: number of evolution paths for i-th inner-ES

Output: $\mathbf{x}^*$: the best-so-far solution
f^*: the best-so-far fitness (cost)

while the *maximal* runtime is not reached **do**
  ▷ *do a parallel for loop over inner-ESs (LM-CMAs)* ◁
  for $i = 1$ to λ' **do**
    if $i <= \mu'$ **then** ▷ *use elitist for part inner-ESs*
      $\mathbf{m}_i, \sigma_i, \mathbf{P}_i, \mathbf{x}_i^*, f_i^* \leftarrow$ **LM-CMA**$(\mathbf{m}_{i;\lambda}, \sigma_{i;\lambda}, \mathbf{P}_{i;\lambda}, \tau)$
    else ▷ *on the multi-recombination strategy*
      if $i <= \mu' + (\lambda' - \mu')/5$ **then** ▷ *for Meta-ES*
        $\sigma \leftarrow U(0.3\sigma', 3.3\sigma')$ ▷ *mutate step-size*
      else ▷ *for step-size diversity*
        $\sigma \leftarrow U(0, \sigma_{\max})$ ▷ *uniformly sample*
      $n_i^e \leftarrow U(n_{\min}^e, n_{\max}^e)$ ▷ *uniformly sample*
      $\mathbf{m}_i, \sigma_i, \mathbf{P}_i, \mathbf{x}_i^*, f_i^* \leftarrow$ **LM-CMA**$(\mathbf{m}', \sigma, \mathbf{P}', \tau, n_i^e)$
  $\mathbf{m}' \leftarrow \sum_{i=1}^{\mu'} w'_{i;\lambda'} \mathbf{m}_{i;\lambda'}$ ▷ *update distribution mean*
  $\sigma' \leftarrow \sum_{i=1}^{\mu'} \frac{w'_{i;\lambda'} \sigma_{i;\lambda'}}{\sqrt{\sum_{i=1}^{u'} w'_{i;\lambda'}}}$ ▷ *multi-recombine for STA*
  ▷ *collective learning on distributed populations* ◁
  $\mathbf{P}' \leftarrow \mathbf{0}_{[\max_{i=1,\dots,\mu'}(n_i^e) \times n]}$ ▷ max *for shape alignment*
  for $i = 1, \dots, \mu'$ **do** ▷ *only consider elitist*
    $\mathbf{P}'[-n_i^e :] += \frac{w'_{i;\lambda'}}{\sqrt{\sum_{i=1}^{u'} w'_{i;\lambda'}}} \mathbf{P}_i[-n_i^e :]$
  $\mathbf{x}^* \leftarrow \min(\mathbf{x}^*, \mathbf{x}_1^*, ..., \mathbf{x}_{\lambda'}^*)$ ▷ *update the best solution*
  $f^* \leftarrow \min(f^*, f_1^*, ..., f_{\lambda'}^*)$ ▷ *update the best fitness*

existing distributed computing systems such as MPI [144], P2P [145], [146], MapReduce [15], [147], Spark [148], BlockChain [149], *ray* provides a flexible programming interface for Python and a powerful distributed scheduling strategy to cater to modern challenging AI applications such as population-based training [150], [151], AutoML [152], and open-ended learning/evolution [153]. Owing to the intrinsic complexity of distributed algorithms, we provide an open-source Python implementation for our proposed meta-framework available at https://github.com/Evolutionary-Intelligence/M-DES, in order to ensure repeatability and benchmarking [154].

For simplicity and ease to analyze, our meta-framework uses the *generational* (rather than *steady-state*) population update strategy at the outer-ES level. Although typically the steady-state method could maximize the parallelism level especially for heterogeneous environments, the asynchronous manner makes distributed black-box optimizers often difficult to debug. In this paper, we consider only the generational population update manner, since it makes the updates and communications of search/mutation distributions easier to understand and analyze under distributing computing.

TABLE I
A SET OF 13 BENCHMARKING FUNCTIONS

	Name	Expression
unimodal (local)	Sphere	$f(x) = \sum_{i=1}^n x_i^2$
	Cigar	$f(x) = x_1^2 + 10^6 \sum_{i=2}^n x_i^2$
	Discus	$f(x) = 10^6 x_1^2 + \sum_{i=2}^n x_i^2$
	Ellipsoid	$f(x) = \sum_{i=1}^n 10^{\frac{6(i-1)}{n-1}} x_i^2$
	DifferentPowers	$f(x) = \sum_{i=1}^n \lvert x_i \rvert^{\frac{2+4(i-1)}{n-1}}$
	Schwefel221	$f(x) = \max(\lvert x_1 \rvert, \cdots, \lvert x_n \rvert)$
	Step	$f(x) = \sum_{i=1}^n (\lfloor x_i + 0.5 \rfloor)^2$
	Schwefel12	$f(x) = \sum_{i=1}^n (\sum_{j=1}^i x_j)^2$
multimodal (global)	Ackley	$f(x) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^n \cos(2\pi x_i)} + 20 + e$
	Rastrigin	$f(x) = 10n + \sum_{i=1}^n (x_i^2 - 10\cos(2\pi x_i))$
	Michalewicz	$f(x) = -\sum_{i=1}^n \sin(x_i)(\sin(\frac{ix_i^2}{\pi}))^{20} + 600$
	Salomon	$f(x) = 1 - \cos(2\pi\sqrt{\sum_{i=1}^n x_i^2}) + 0.1\sqrt{\sum_{i=1}^n x_i^2}$
	ScaledRastrigin	$f(x) = 10n + \sum_{i=1}^n ((10^{\frac{i-1}{n-1}} x_i)^2 - 10\cos(2\pi 10^{\frac{i-1}{n-1}} x_i))$

IV. Large-Scale Numerical Experiments

To study the benefits (and costs) of our proposed meta-framework (simply named as DLMCMA here), we conduct numerical experiments on a set of large-scale benchmarking functions with memory-expensive fitness evaluations, arguably reflecting many challenging data-driven optimization problems. To ensure repeatability and promote benchmarking [155], all involved experimental data and Python code are openly available at our companion website https://github.com/Evolutionary-Intelligence/M-DES.

A. Experimental Settings

Test Functions: We choose a set of 13 commonly used test functions, as shown in Table I for their mathematical formula (see e.g., COCO/BBOB [22] or NeverGrad [156] for implementations). These functions can be roughly classified as two families (i.e., unimodal and multimodal functions) to compare *local* and *global* search abilities, respectively. For benchmarking large-scale BBO, the dimensions of all the test functions are set to 2000. We also use the standard angle-preserving (i.e., rotation) transformation [17] (rather than [24]) and random shift to generate non-separability and avoid the origin as the global optimum, respectively. This involved matrix-vector multiplication operator results in the memory-expensive fitness evaluation, arguably one significant feature of many real-world data-driven optimization problems. In order to speedup parallel fitness evaluations, we use the simple yet efficient *shared memory* trick for our distributed optimizer.

Benchmarking Optimizers: To benchmark the advantages and disadvantages of different approaches, we select a total of 61 black-box optimizers from different families (i.e., Evolution Strategies - ES, Natural Evolution Strategies - NES, Estimation of Distribution Algorithms - EDA, Cross-Entropy Method - CEM, Differential Evolution - DE, Particle Swarm Optimizer - PSO, Cooperative Coevolution - CC, Simulated Annealing - SA, Genetic Algorithms - GA, Evolutionary Programming - EP, Pattern Search - PS, and Random Search - RS) implemented in a recently well-designed Python library called **PyPop7**. Due to the page limit, here we only use their *abbreviated*

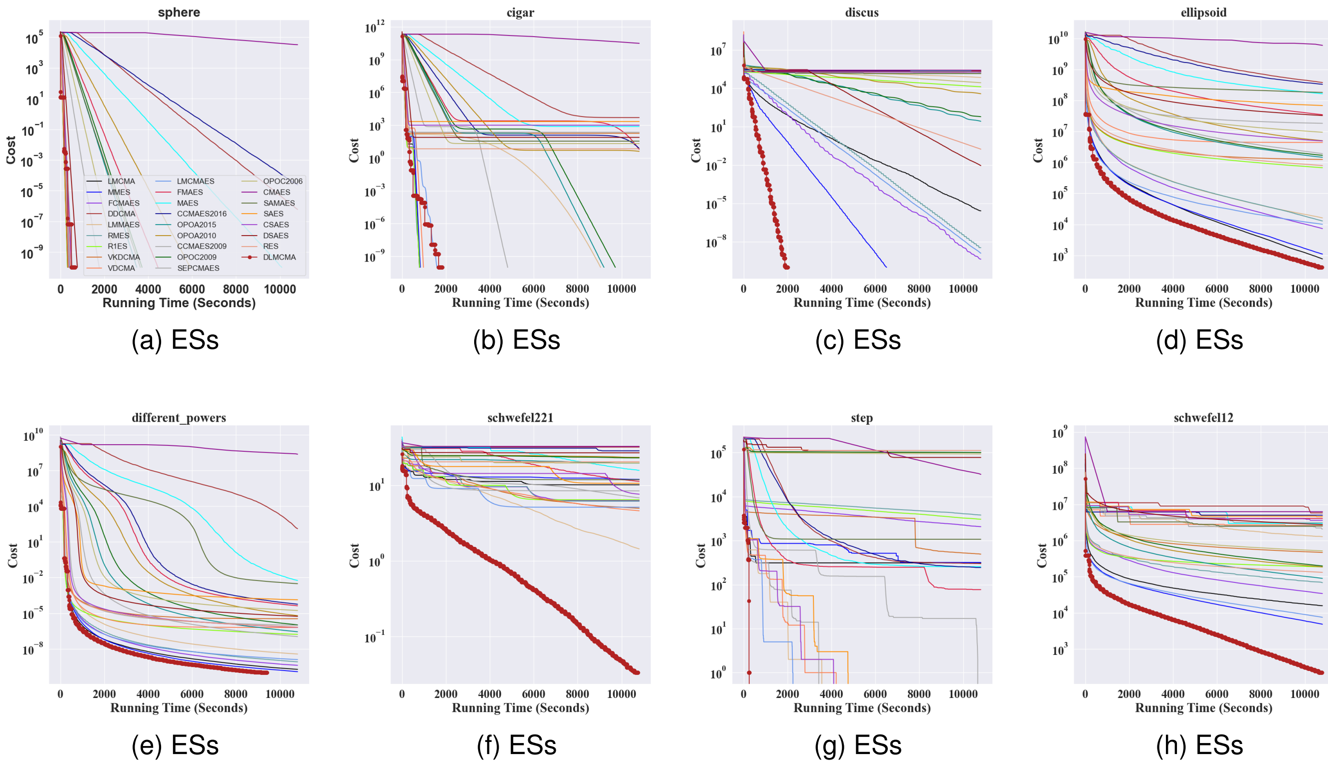


Fig. 1. Median convergence curves on a set of 2000-d *unimodal* functions given the maximal runtime (3 hours) and the cost threshold ($1e^{-10}$).

(rather *full*) names to avoid legend confusion when plotting the convergence figures. For implementation details about these optimizers and their hyper-parameter settings, please refer to this online website pypop.rtfd.io and references therein. Overall, these optimizers are grouped into two classes for plotting convergence curves: ES-based optimizers and others, as shown in Fig. (1, 3) and Fig. (2, 4), respectively.

Computing Environments: We set the parallel number of inner-ESs (λ') to 380 and 540 (CPU cores) for our DLM-CMA on unimodal and multimodal functions, respectively. On multimodal functions, more parallel inner-ESs often bring more population diversity. Although the optimal setting of λ' is problem-dependent, we do not fine-tune it for each function. We empirically set the isolation time τ to 150 seconds on all functions though this is *not necessarily* an optimal value for each function.

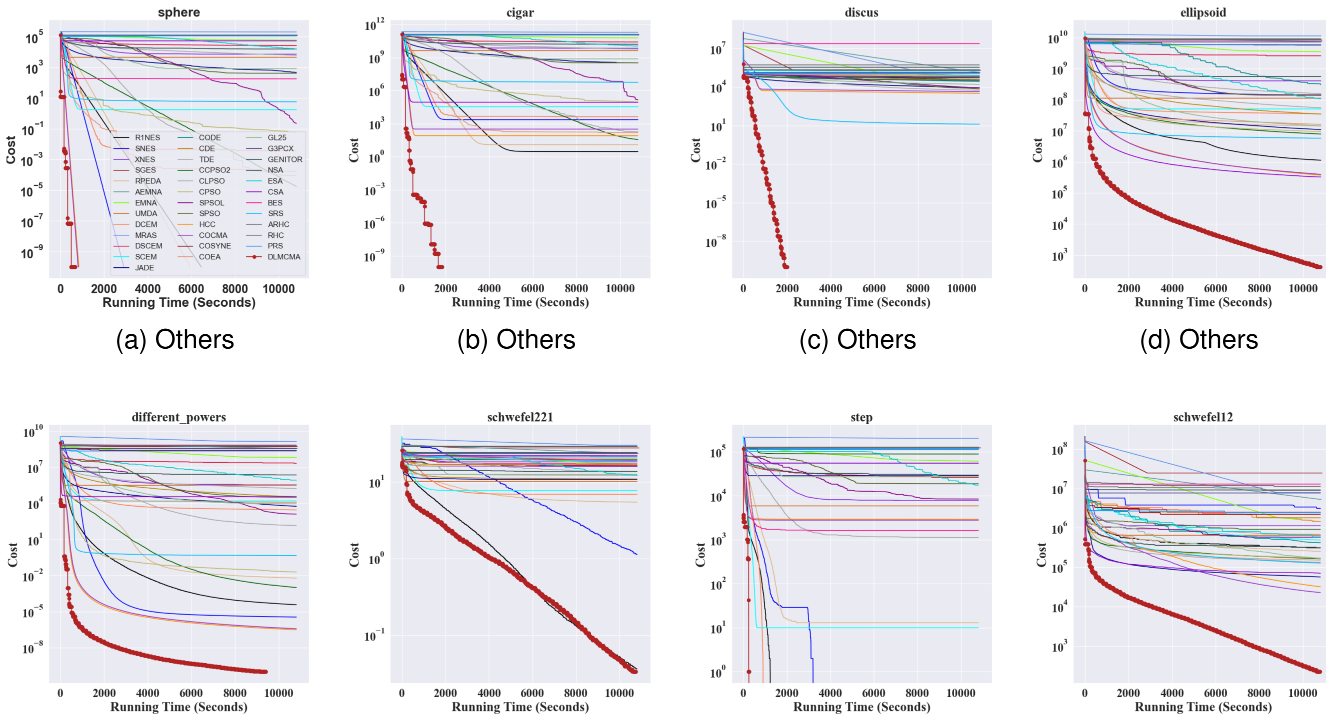
Owing to the time-consuming experiment process for LSO, We run each optimizer 10 and 4 times on each unimodal and multimodal function, respectively. The total CPU single-core runtime needed in our experiments is estimated up to **18600 hours**, that is, 775 days = $(10 \times 8 \times 3 + 4 \times 5 \times 3) \times 62$ hours.

B. Comparing Local Search Abilities

On the *sphere* function, arguably one of the simplest test cases for continuous optimization, it is highly expected that the optimizer could obtain a fast convergence. Most ES-based optimizers could obtain a satisfactory (not necessarily optimal) performance except some CMA-ES variants with quadratic complexity (Fig. 1a). For quadratic-complexity CMA-ES variants (e.g., DD-CMA [117], MA-ES [120], and C-CMA-ES [122]), the overall runtime is dominated heavily by the CMA mechanism rather than the function evaluation time, therefore resulting in a much slow adaptation speed.

There is one predominated search direction needed to explore for the *cigar* function. This means that a low-rank learning strategy (e.g., R1-ES [114]) is typically enough to capture the main direction via adaptation. Owing to the extra cost brought from distributed computing, our meta-framework (DLMCMA) obtains a slightly slow convergence speed as shown in Fig. 1b (reflecting limitations of parallelism [157]). However, it could approximate the low-rank learning ability well, given that the initial number of reconstructed evolution paths does not match the optimal setting.

For both functions *discus* and *ellipsoid*, there exist multiple promising search directions (see their relatively even eigen-value distributions). Therefore, a much richer reconstruction model is preferred for CMA. On the *discus* function (Fig. 1c), our DLMCMA could show the >3x runtime speed w.r.t. the second ranked optimizer (i.e., MM-ES [119]). On the *ellipsoid* function, our DLMCMA nearly always shows the best convergence speed during evolution (Fig. 1d), because its collective learning strategy maintains the better diversity of reconstructed evolution paths via utilizing the distributed computing resource. Interestingly, similar observations could also be found in another two challenging functions (i.e., *differentpowers* and *schwefel12*) with multiple search directions (as shown in Fig. 1e and 1f, respectively).



(a) Others (b) Others (c) Others (d) Others

(e) Others (f) Others (g) Others (h) Others

Fig. 2. Median convergence curves on a set of 2000-d *unimodal* functions given the maximal runtime (3 hours) and the cost threshold ($1e^{-10}$).

For both *schwefel221* and *step* functions, there are a large number of plateaus in high-dimensional cases, which result in a rugged fitness landscape. For ESs, a key challenge is to properly adapt the global step-size to pass these plateaus[14]. Luckily, our meta-framework can tackle this challenge well and can obtain the best convergence rate on both of them (Fig. 1g and 1h), since our STA strategy could keep the diversity of the global step-size well while avoiding to diverge it (with the help of the elitist strategy)[15].

As we can clearly see from Fig. 3, our DLMCMA could always obtain the best convergence rate when compared with all other algorithm families except that on *schwefel221* R1-NES shows a very similar performance. Overall, our DLMCMA can obtain the best or competitive performance on these unimodal function, empirically validating the benefits from multilevel distributed learning.

C. Comparing Global Search Abilities

For minimizing the *ackley* function, it seems to be looking for a needle in the haystack. However, there is a global landscape structure to be available, which can be used to accelerate the global convergence rate of the optimizer (if well-utilized). We find that many large-scale variants of CMA-ES utilize this property, even given a small population setting. Our DLMCMA can approximate this global structure well and therefore it obtains the best convergence rate after bypassing all (shallow) local optima (Fig. 3a and 4a).

On the classic *rastrigin* function, there exist a large number of relatively deep local minima, which can hinder the optimization process. To escape from these local optima, the simple yet efficient restart [159] strategy from CMA-ES will increase the number of offspring after each restart. Clearly, our DLMCMA obtains much better results among all optimizers, with the help of multiple restarts (Fig. 3b and 4b).

Our proposed DLMCMA obtains the second ranking only after UMDA [160] on the *michalewicz* function (Fig. 3c and 4c), which seems to have a relatively weak global structure. The default population size of UMDA is relatively large (200) while that of each (local) LM-CMA used in our DLMCMA is very small (19) by default. Despite this difference, our DLMCMA still can drive the parallel evolution process over structured populations to approach the best after 3 hours.

On the multimodal function *salomon*, our D-LM-CMA finds the best solution much faster than all other optimizers (Fig. 3d and 4d). However, the restart strategy cannot help to find a better solution, which may means that this found solution is near a deep local optimum. On another multimodal function *scaledrastrigin*, our DLMCMA ranks the third, only after R1-NES and SNES [89] (Fig. 3e and 4e). We notice that the parallel evolution process stagnates even at the early stage, which means that we need a better restart strategy for this function. We leave it for future work.

In summary, our meta-framework achieves the very com-

[14]In other worlds, this needs to find an appropriate *evolution window*, popularized by Rechenberg (one of the evolutionary computation pioneers).

[15]The punctuated-equilibria-style convergence is dated back to at least [158], depending on the used viewpoint (as pointed out by Schwefel, one of the evolutionary computation pioneers).

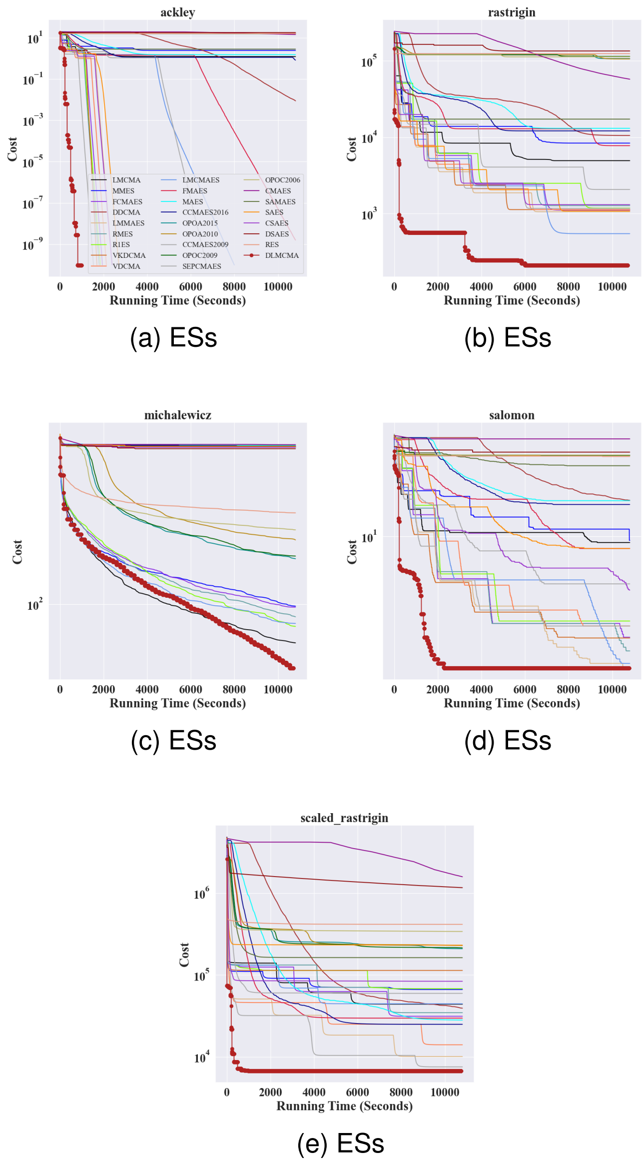


Fig. 3. Median convergence curves on a set of 2000-d *multimodal* functions given the maximal runtime (3 hours) and the cost threshold ($1e^{-10}$).

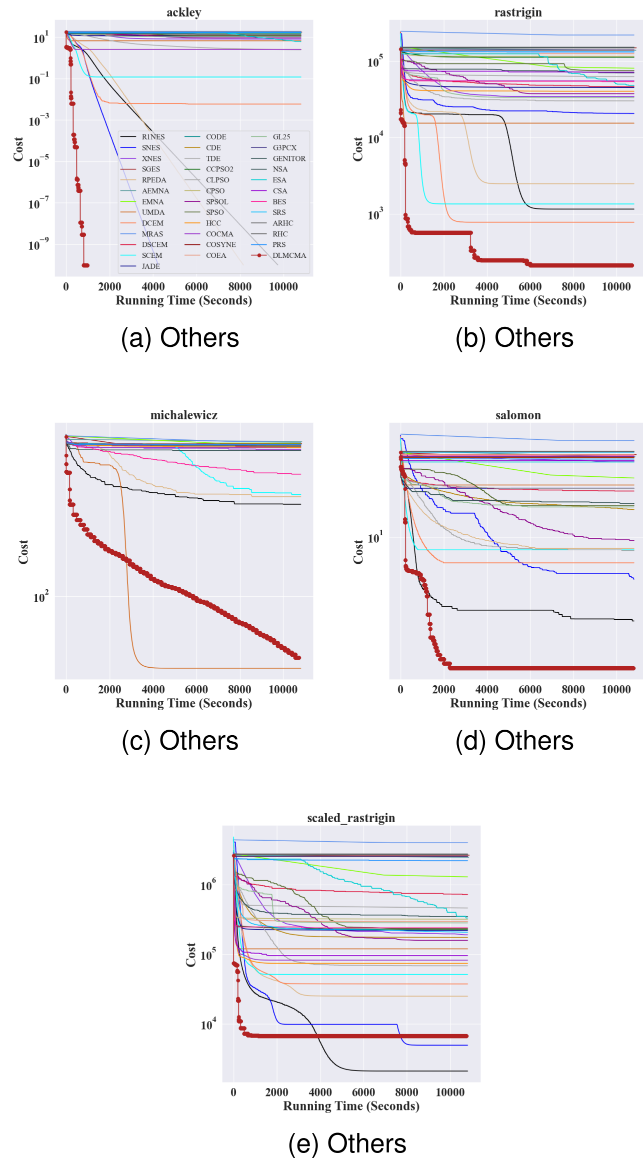


Fig. 4. Median convergence curves on a set of 2000-d *multimodal* functions given the maximal runtime (3 hours) and the cost threshold ($1e^{-10}$).

petitive performance on both unimodal and multimodal test functions considered in the experiments, under the challenging distributed computing scenarios.

V. Conclusion

In this paper, we propose a multilevel learning-based meta-framework to parallelize one large-scale variant of CMA-ES called LM-CMA, significantly extending our previous conference paper [161]. Within this meta-framework, four main design choices are made to control distribution mean update, global step-size adaptation, and CMA reconstruction for effectiveness and efficiency. A large number of comparative experiments show the benefits (and costs) of our proposed meta-framework.

In principle, the proposed distributed meta-framework can be integrated into some other meta-heuristics [162] with more or less modifications.

References

[1] G. E. Moore, "Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp. 114 ff." *IEEE Solid-State Circuits Society Newsletter*, vol. 11, no. 3, pp. 33–35, 2006.

[2] ——, "Cramming more components onto integrated circuits," *PIEEE*, vol. 86, no. 1, pp. 82–85, 1998.

[3] M. Bohr, "A 30 year retrospective on Dennard's MOSFET scaling paper," *IEEE Solid-State Circuits Society Newsletter*, vol. 12, no. 1, pp. 11–13, 2007.

[4] W. Liu, F. Lombardi, and M. Shulte, "A retrospective and prospective view of approximate computing," *PIEEE*, vol. 108, no. 3, pp. 394–399, 2020.

[5] J. Shalf, "The future of computing beyond Moore's law," *Philosophical Transactions of the Royal Society A*, vol. 378, no. 2166, p. 20190061, 2020.

[6] C. E. Leiserson, N. C. Thompson *et al.*, "There's plenty of room at the top: What will drive computer performance after moore's law?" *Science*, vol. 368, no. 6495, p. eaam9744, 2020.

[7] J. L. Hennessy and D. A. Patterson, "A new golden age for computer architecture," *CACM*, vol. 62, no. 2, pp. 48–60, 2019.

[8] M. D. Hill and M. R. Marty, "Amdahl's law in the multicore era," *Computer*, vol. 41, no. 7, pp. 33–38, 2008.

[9] D. B. Fogel, *Evolutionary computation: Toward a new philosophy of machine intelligence*, 3rd ed. John Wiley & Sons, 2006.

[10] J. H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. MIT Press, 1992.

[11] T. Back, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: Comments on the history and current state," *TEVC*, vol. 1, no. 1, pp. 3–17, 1997.

[12] A. E. Eiben and J. Smith, "From evolutionary computation to the evolution of things," *Nature*, vol. 521, no. 7553, pp. 476–482, 2015.

[13] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *ECJ*, vol. 1, no. 1, pp. 1–23, 1993.

[14] J. Fan, S. Shen *et al.*, "A high-resolution summary of cambrian to early triassic marine invertebrate biodiversity," *Science*, vol. 367, no. 6475, pp. 272–277, 2020.

[15] F. Ferrucci, P. Salza, and F. Sarro, "Using hadoop mapreduce for parallel genetic algorithms: A comparison of the global, grid and island models," *ECJ*, vol. 26, no. 4, pp. 535–567, 2018.

[16] N. Hansen, "The cma evolution strategy: A tutorial," 2023. [Online]. Available: https://arxiv.org/abs/1604.00772

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

[18] ——, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation," in *CEC*, 1996, pp. 312–317.

[19] I. Loshchilov, "LM-CMA: An alternative to L-BFGS for large-scale black box optimization," *ECJ*, vol. 25, no. 1, pp. 143–171, 2017.

[20] I. Loshchilov, T. Glasmachers, and H. Beyer, "Large scale black-box optimization by limited-memory matrix adaptation," *TEVC*, vol. 23, no. 2, pp. 353–358, 2019.

[21] S. Kern, S. D. Müller, N. Hansen, D. Büche, J. Ocenasek, and P. Koumoutsakos, "Learning probability distributions in continuous evolutionary algorithms – a comparative review," *Natural Computing*, vol. 3, no. 1, pp. 77–112, 2004.

[22] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík, "Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009," in *GECCO*. Portland, Oregon, USA: ACM, 2010, pp. 1689–1696.

[23] N. Hansen, A. Auger, D. Brockhoff, and T. Tušar, "Anytime performance assessment in blackbox optimization benchmarking," *TEVC*, vol. 26, no. 6, pp. 1293–1305, 2022.

[24] K. Varelas, O. A. El Hara, D. Brockhoff, N. Hansen, D. M. Nguyen, T. Tušar, and A. Auger, "Benchmarking large-scale continuous optimizers: The bbob-largescale testbed, a coco software guide and beyond," *Applied Soft Computing*, vol. 97, p. 106737, 2020.

[25] A. Auger and N. Hansen, "A SIGEVO impact award for a paper arising from the coco platform: A summary and beyond," *SIGEVOlution*, vol. 13, no. 4, 2021.

[26] M. Bujny, N. Aulig, M. Olhofer, and F. Duddeck, "Identification of optimal topologies for crashworthiness with the evolutionary level set method," *International Journal of Crashworthiness*, vol. 23, no. 4, pp. 395–416, 2018.

[27] J. Zhang, P. Fiers, K. A. Witte, R. W. Jackson, K. L. Poggensee, C. G. Atkeson, and S. H. Collins, "Human-in-the-loop optimization of exoskeleton assistance during walking," *Science*, vol. 356, no. 6344, pp. 1280–1284, 2017.

[28] M. Schoenauer, R. Akrour, M. Sebag, and J.-C. Souplet, "Programming by feedback," in *ICML*, E. P. Xing and T. Jebara, Eds., vol. 32. Beijing, China: PMLR, 2014, pp. 1503–1511.

[29] R. T. Lange, T. Schaul, Y. Chen, T. Zahavy, V. Dalibard, C. Lu, S. Singh, and S. Flennerhag, "Discovering evolution strategies via meta-black-box optimization," in *ICLR*, Kigali Rwanda, 2023.

[30] G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, and T. Vicsek, "Optimized flocking of autonomous drones in confined environments," *Science Robotics*, vol. 3, no. 20, p. eaat3536, 2018.

[31] H. Ma, A. Narayanaswamy, P. Riley, and L. Li, "Evolving symbolic density functionals," *Science Advances*, vol. 8, no. 36, p. eabq0279, 2022.

[32] M. D. Hill, "What is scalability?" *ACM SIGARCH Computer Architecture News*, vol. 18, no. 4, pp. 18–21, 1990.

[33] Y. Ollivier, L. Arnold, A. Auger, and N. Hansen, "Information-geometric optimization algorithms: A unifying picture via invariance principles," *JMLR*, vol. 18, no. 18, pp. 1–65, 2017.

[34] N. Hansen, "Invariance, self-adaptation and correlated mutations in evolution strategies," in *PPSN*, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds., vol. 1917. Paris, France: Springer Berlin Heidelberg, 2000, pp. 355–364.

[35] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies – a comprehensive introduction," *Natural Computing*, vol. 1, pp. 3–52, 2002.

[36] N. Hansen and A. Auger, "Principled design of continuous stochastic search: From theory to practice," in *Theory and Principled Methods for the Design of Metaheuristics*. Springer, 2014, pp. 145–180.

[37] V. Vanchurin, Y. I. Wolf, M. I. Katsnelson, and E. V. Koonin, "Toward a theory of evolution as multilevel learning," *PNAS*, vol. 119, no. 6, p. e2120037119, 2022.

[38] V. Vanchurin, Y. I. Wolf, E. V. Koonin, and Mikhail I. Katsnelson, "Thermodynamics of evolution and the origin of life," *PNAS*, vol. 119, no. 6, p. e2120042119, 2022.

[39] D. V. Arnold and A. MacLeod, "Step length adaptation on ridge functions," *ECJ*, vol. 16, no. 2, pp. 151–184, 2008.

[40] H.-G. Beyer and M. Hellwig, "Controlling population size and mutation strength by meta-es under fitness noise," in *FOGA*. Adelaide, Australia: ACM, 2013, pp. 11–24.

[41] E. Jonas, Q. Pu, S. Venkataraman, I. Stoica, and B. Recht, "Occupy the cloud: Distributed computing for the 99%," in *Proceedings of Symposium on Cloud Computing*. Santa Clara California: ACM, 2017, pp. 445–451.

[42] D. Brookes, A. Busia, C. Fannjiang, K. Murphy, and J. Listgarten, "A view of estimation of distribution algorithms through the lens of expectation-maximization," in *GECCO*. Cancún, Mexico: ACM, 2020, pp. 189–190.

[43] H.-P. Schwefel, "Evolutionary learning optimum-seeking on parallel computer architectures," in *Systems Analysis and Simulation I: Theory and Foundations*, A. Sydow, S. G. Tzafestas, and R. Vichnevetsky, Eds., vol. 1. Springer, 1988, pp. 217–225.

[44] D. V. Arnold, "Weighted multirecombination evolution strategies," *TCS*, vol. 361, no. 1, pp. 18–37, 2006.

[45] G. Rudolph, "On correlated mutations in evolution strategies," in *PPSN*, R. Männer and B. Manderick, Eds. Brussels, Belgium: Elsevier, 1992, pp. 107–116.

[46] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *JMLR*, vol. 13, pp. 281–305, 2012.

[47] J. Heinerman, J. Stork, M. A. R. Coy, J. Hubert, A. E. Eiben, T. Bartz-Beielstein, and E. Haasdijk, "Is social learning more than parameter tuning?" in *GECCO Companion*. Berlin, Germany: ACM, 2017, pp. 63–64.

[48] H.-P. Schwefel, "Collective intelligence in evolving systems," in *Ecodynamics*, W. Wolff, C.-J. Soeder, and F. R. Drepper, Eds. Fed. Rep. of Germany: Springer Berlin, Heidelberg, 1988, pp. 95–100.

[49] F. McSherry, M. Isard, and D. G. Murray, "Scalability! but at what cost?" in *HOTOS*. Switzerland: USENIX Association, 2015, p. 14.

[50] N. Hansen, "The CMA evolution strategy: A comparing review," in *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*, J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 75–102.

[51] T. Bäck, C. Foussette, and P. Krause, *Contemporary evolution strategies*, 1st ed. Springer Berlin Heidelberg, 2013.

[52] G. Rudolph, "Evolutionary strategies," in *Handbook of Natural Computing*, G. Rozenberg, T. Bäck, and J. N. Kok, Eds. Springer Berlin Heidelberg, 2012, pp. 673–698.

[53] M. Emmerich, O. M. Shir, and H. Wang, "Evolution strategies," in *Handbook of Heuristics*, R. Martí, P. M. Pardalos, and M. G. C. Resende, Eds. Cham: Springer International Publishing, 2018, pp. 89–119.

[54] M. Armbrust, A. Fox, and et al., "A view of cloud computing," *CACM*, vol. 53, no. 4, pp. 50–58, 2010.

[55] S. Chasins, A. Cheung *et al.*, "The sky above the clouds," 2022. [Online]. Available: https://arxiv.org/abs/2205.07147

[56] D. Reed, D. Gannon, and J. Dongarra, "HPC forecast: Cloudy and uncertain," *CACM*, vol. 66, no. 2, pp. 82–90, 2023.

[57] M. Schoenauer, "Evolutionary algorithms," in *Handbook of Evolutionary Thinking in the Sciences*, T. Heams, P. Huneman, G. Lecointre, and M. Silberstein, Eds. Dordrecht: Springer Netherlands, 2015, pp. 621–635.

[58] D. C. Fisher, "Your favorite parallel algorithms might not be as fast as you think," *TC*, vol. 37, no. 02, pp. 211–213, 1988.

[59] D. H. Wolpert, "What is important about the no free lunch theorems?" in *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*, P. M. Pardalos, V. Rasskazova, and M. N. Vrahatis, Eds. Cham: Springer International Publishing, 2021, pp. 373–388.

[60] W. G. Macready, A. G. Siapas, and S. A. Kauffman, "Criticality and parallelism in combinatorial optimization," *Science*, vol. 271, no. 5245, pp. 56–59, 1996.

[61] J. Dean, D. Patterson, and C. Young, "A new golden age in computer architecture: Empowering the machine-learning revolution," *IEEE Micro*, vol. 38, no. 2, pp. 21–29, 2018.

[62] S. Hooker, "The hardware lottery," *CACM*, vol. 64, no. 12, pp. 58–65, 2021.

[63] G. Rudolph, "Parallel evolution strategies," in *Parallel Metaheuristics: A New Class of Algorithms*. John Wiley & Sons, Inc., 2005, pp. 155–169.

[64] U. Bernutat-Buchmann and J. Krieger, "Evolution strategies in numerical optimization on vector computers," in *ICPC*, vol. 83, 1983, pp. 99–105.

[65] F. Hoffmeister, "Scalable parallelism by evolutionary algorithms," in *Parallel Computing and Mathematical Optimization*, M. Grauer and D. B. Pressmar, Eds., vol. 367. Siegen, FRG: Springer Berlin, Heidelberg, 1991, pp. 177–198.

[66] C. Kappler, T. Bäck, J. Heistermann, A. Van de Velde, and M. Zamparelli, "Refueling of a nuclear power plant: Comparison of a naive and a specialized mutation operator," in *PPSN*, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds., vol. 1141. Berlin, Germany: Springer Berlin Heidelberg, 1996, pp. 829–838.

[67] T. Bäck, "Parallel optimization of evolutionary algorithms," in *PPSN*, Y. Davidor, H.-P. Schwefel, and R. Männer, Eds., vol. 866. Jerusalem, Israel: Springer, 1994, pp. 418–427.

[68] D. Keller, "Optimization of ply angles in laminated composite structures by a hybrid, asynchronous, parallel evolutionary algorithm," *Composite Structures*, vol. 92, no. 11, pp. 2781–2790, 2010.

[69] P. Eberhard, F. Dignath, and L. Kübler, "Parallel evolutionary optimization of multibody systems with application to railway dynamics," *Multibody System Dynamics*, vol. 9, pp. 143–164, 2003.

[70] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities, reprinted from the afips conference proceedings, vol. 30 (atlantic city, nj, apr. 18–20), afips press, reston, va., 1967, pp. 483–485, when dr. amdahl was at international business machines corporation, sunnyvale, california," *IEEE Solid-State Circuits Society Newsletter*, vol. 12, no. 3, pp. 19–20, 2007.

[71] ——, "Computer architecture and amdahl's law," *Computer*, vol. 46, no. 12, pp. 38–46, 2013.

[72] S. D. Müller, N. Hansen, and P. Koumoutsakos, "Increasing the serial and the parallel performance of the CMA-evolution strategy with large populations," in *PPSN*, J. J. M. Guervós, P. Adamidis, H.-G. Beyer, H.-P. Schwefel, and J.-L. Fernández-Villacañas, Eds., vol. 2439. Granada, Spain: Springer Berlin Heidelberg, 2002, pp. 422–431.

[73] G. Rudolph, "Global optimization by means of distributed evolution strategies," in *PPSN*, H.-P. Schwefel and R. Männer, Eds., vol. 496. Dortmund, Germany: Springer, 1990, pp. 209–213.

[74] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *ECJ*, vol. 11, no. 1, pp. 1–18, 2003.

[75] F. Biscani and D. Izzo, "A parallel global multiobjective framework for optimization: Pagmo," *JOOS*, vol. 5, no. 53, p. 2338, 2020.

[76] X. He, Z. Zheng, C. Chen, Y. Zhou, C. Luo, and Q. Lin, "Distributed evolution strategies for black-box stochastic optimization," *TPDS*, vol. 33, no. 12, pp. 3718–3731, 2022.

[77] F. Neumann, P. S. Oliveto, G. Rudolph, and D. Sudholt, "On the effectiveness of crossover for migration in parallel evolutionary algorithms," in *GECCO*. Dublin, Ireland: ACM, 2011, pp. 1587–1594.

[78] S. Garg, K. Shiragur, D. M. Gordon, and M. Charikar, "Distributed algorithms from arboreal ants for the shortest path problem," *PNAS*, vol. 120, no. 6, p. e2207959120, 2023.

[79] P. K. Lehre and D. Sudholt, "Parallel black-box complexity with tail bounds," *TEVC*, vol. 24, no. 6, pp. 1010–1024, 2020.

[80] A. Mambrini, D. Sudholt, and X. Yao, "Homogeneous and heterogeneous island models for the set cover problem," in *PPSN*, C. A. C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, Eds., vol. 7491. Taormina, Italy: Springer Berlin Heidelberg, 2012, pp. 11–20.

[81] C. Qian, "Distributed pareto optimization for large-scale noisy subset selection," *TEVC*, vol. 24, no. 4, pp. 694–707, 2020.

[82] G. Rudolph, "Massively parallel simulated annealing and its relation to evolutionary algorithms," *ECJ*, vol. 1, no. 4, pp. 361–383, 1993.

[83] D. Wilson, K. Veeramachaneni, and U.-M. O'Reilly, "Cloud scale distributed evolutionary strategies for high dimensional problems," in *Applications of Evolutionary Computation*, A. I. Esparcia-Alcázar, Ed., vol. 7835. Vienna, Austria: Springer Berlin Heidelberg, 2013, pp. 519–528.

[84] C. L. Mueller, B. Baumgartner, G. Ofenbeck, B. Schrader, and I. F. Sbalzarini, "PCMALib: A parallel fortran 90 library for the evolution strategy with covariance matrix adaptation," in *GECCO*. Montreal, Québec, Canada: ACM, 2009, pp. 1411–1418.

[85] C. Rossant, B. Fontaine, and D. F. Goodman, "Playdoh: A lightweight python library for distributed computing and optimisation," *Journal of Computational Science*, vol. 4, no. 5, pp. 352–359, 2013.

[86] P. Incardona, A. Leo, Y. Zaluzhnyi, R. Ramaswamy, and I. F. Sbalzarini, "OpenFPM: A scalable open framework for particle and particle-mesh codes on parallel computers," *Computer Physics Communications*, vol. 241, pp. 155–177, 2019.

[87] D. Hakkarinen, T. Camp, Z. Chen, and A. Haas, "Reduced data communication for parallel CMA-ES for reacts," in *PDP*. Munich, Germany: IEEE, 2012, pp. 97–101.

[88] T. Glasmachers, "A natural evolution strategy with asynchronous strategy updates," in *GECCO*. Amsterdam, The Netherlands: ACM, 2013, pp. 431–438.

[89] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, "Natural evolution strategies," *JMLR*, vol. 15, no. 27, pp. 949–980, 2014.

[90] L. Jostins and J. Jaeger, "Reverse engineering a gene network using an asynchronous parallel evolution strategy," *BMC Systems Biology*, vol. 4, pp. 1–16, 2010.

[91] Y. Fomekong-Nanfack, J. A. Kaandorp, and J. Blom, "Efficient parameter estimation for spatio-temporal models of pattern formation: Case study of drosophila melanogaster," *Bioinformatics*, vol. 23, no. 24, pp. 3356–3363, 2007.

[92] G. Cucu, L. Rolshoven, F. Vorpe, P. Cudré-Mauroux, and T. Glasmachers, "Dibb: Distributing black-box optimization," in *GECCO*. Boston, Massachusetts: ACM, 2022, pp. 341–349.

[93] M. Porcelli and P. L. Toint, "Exploiting problem structure in derivative free optimization," *ACM-TOMS*, vol. 48, no. 1, pp. 1–25, 2022.

[94] Q. Duan, C. Shao, G. Zhou, H. Yang, Q. Zhao, and Y. Shi, "Cooperative coevolution for non-separable large-scale black-box optimization: Convergence analyses and distributed accelerations," 2023.

[95] A. Kucharavy, M. Monti, R. Guerraoui, and L. Dolamic, "Byzantine-resilient learning beyond gradients: Distributing evolutionary search," in *GECCO*. Lisbon, Portugal: ACM, 2023, p. in press.

[96] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency: The Works of Leslie Lamport*. New York, NY, USA: ACM, 2019, pp. 203–226.

[97] F. Rehbach, M. Zaefferer, A. Fischbach, G. Rudolph, and T. Bartz-Beielstein, "Benchmark-driven configuration of a parallel model-based optimization algorithm," *TEVC*, vol. 26, no. 6, pp. 1365–1379, 2022.

[98] I. Rechenberg, "Evolutionsstrategien," in *Simulationsmethoden in der Medizin und Biologie*, S. Koller, P. L. Reichertz, K. Überla, J. Anderson, G. Goos, F. Gremy, H.-J. Jesdinsky, H.-J. Lange, B. Schneider, G. Segmüller, G. Wagner, B. Schneider, and U. Ranft, Eds., vol. 8. Hannover: Springer Berlin Heidelberg, 1978, pp. 83–114.

[99] ——, *Evolutionsstrategie 94*. Frommann-Holzboog, Stuttgart, 1994, vol. 1.

[100] D. V. Arnold and A. S. Castellarin, "A novel approach to adaptive isolation in evolution strategies," in *GECCO*. Montreal, Québec, Canada: ACM, 2009, pp. 491–498.

[101] D. V. Arnold and A. MacLeod, "Hierarchically organised evolution strategies on the parabolic ridge," in *GECCO*. Seattle, Washington, USA: ACM, 2006, pp. 437–444.

[102] H.-G. Beyer, M. Dobler, C. Hämmerle, and P. Masser, "On strategy parameter control by meta-es," in *GECCO*. Montreal, Québec, Canada: ACM, 2009, pp. 499–506.

[103] H.-G. Beyer and M. Hellwig, "Mutation strength control by meta-es on the sharp ridge," in *GECCO*. Philadelphia, Pennsylvania, USA: ACM, 2012, pp. 305–312.

[104] M. Hellwig and H.-G. Beyer, "Mutation strength control via meta evolution strategies on the ellipsoid model," *TCS*, vol. 623, pp. 160–179, 2016.

[105] P. Spettel, H.-G. Beyer, and M. Hellwig, "Steady state analysis of a multi-recombinative meta-ES on a conically constrained problem with comparison to σSA and CSA," in *FOGA*. New York, NY, USA: ACM, 2019, pp. 43–57.

[106] S. Baluja, "The evolution of genetic algorithms: Towards massive parallelism," in *ICML*. Amherst, Massachusetts, USA: Morgan Kaufmann, 1993, pp. 1–8.
[107] G. Rudolph, "Parallel approaches to stochastic global optimization," in *Proceedings of European Workshops on Parallel Computing*. Barcelona, Spain: IOS Press, 1992, pp. 256–267.
[108] K. Weinert, J. Mehnen, and G. Rudolph, "Dynamic neighborhood structures in parallel evolution strategies," *Complex Systems*, vol. 13, no. 3, pp. 227–244, 2001.
[109] R. Wang and J. Zhi, "A hands-on guide to distributed computing paradigms for evolutionary computation," in *GECCO Companion*. Cancún, Mexico: ACM, 2020, pp. 1055–1074.
[110] T. Harada and E. Alba, "Parallel genetic algorithms: A useful survey," *ACM Computing Surveys*, vol. 53, no. 4, pp. 86:1–39, 2020.
[111] K. Varelas, A. Auger *et al.*, "A comparative study of large-scale variants of CMA-ES," in *PPSN*, A. Auger, C. M. Fonseca, N. Lourenço, P. Machado, L. Paquete, and D. Whitley, Eds., vol. 11101. Coimbra, Portugal: Springer International Publishing, 2018, pp. 3–15.
[112] Y. Akimoto and N. Hansen, "Online model selection for restricted covariance matrix adaptation," in *PPSN*, J. Handl, E. Hart, P. R. Lewis, M. López-Ibáñez, G. Ochoa, and B. Paechter, Eds., vol. 9921. Edinburgh, UK: Springer International Publishing, 2016, pp. 3–13.
[113] ——, "Projection-based restricted covariance matrix adaptation for high dimension," in *GECCO*. Denver, Colorado, USA: ACM, 2016, pp. 197–204.
[114] Z. Li and Q. Zhang, "A simple yet efficient evolution strategy for large-scale black-box optimization," *TEVC*, vol. 22, no. 5, pp. 637–646, 2018.
[115] Z. Li, Q. Zhang, X. Lin, and H.-L. Zhen, "Fast covariance matrix adaptation for large-scale black-box optimization," *TCYB*, vol. 50, no. 5, pp. 2073–2083, 2020.
[116] T. Schaul, T. Glasmachers, and J. Schmidhuber, "High dimensions and heavy tails for natural evolution strategies," in *GECCO*. Dublin, Ireland: Association for Computing Machinery, 2011, pp. 845–852.
[117] Y. Akimoto and N. Hansen, "Diagonal acceleration for covariance matrix adaptation evolution strategies," *ECJ*, vol. 28, no. 3, pp. 405–435, 2020.
[118] R. Ros and N. Hansen, "A simple modification in CMA-ES achieving linear time and space complexity," in *PPSN*, G. Rudolph, T. Jansen, N. Beume, S. Lucas, and C. Poloni, Eds., vol. 5199. Dortmund, Germany: Springer Berlin Heidelberg, 2008, pp. 296–305.
[119] X. He, Z. Zheng, and Y. Zhou, "Mmes: Mixture model-based evolution strategy for large-scale optimization," *TEVC*, vol. 25, no. 2, pp. 320–333, 2021.
[120] H.-G. Beyer, "Design principles for matrix adaptation evolution strategies," in *GECCO*. Cancún, Mexico: ACM, 2020, pp. 682–700.
[121] H.-G. Beyer and B. Sendhoff, "Simplify your covariance matrix adaptation evolution strategy," *TEVC*, vol. 21, no. 5, pp. 746–759, 2017.
[122] O. Krause, D. R. Arbonès, and C. Igel, "CMA-ES with optimal covariance update and storage complexity," in *NeurIPS*, Barcelona, Spain, 2016.
[123] O. Krause and C. Igel, "A more efficient rank-one covariance matrix update for evolution strategies," in *FOGA*. Aberystwyth, United Kingdom: ACM, 2015, pp. 129–136.
[124] D. V. Arnold and N. Hansen, "Active covariance matrix adaptation for the (1+1)-CMA-ES," in *GECCO*. Portland, Oregon, USA: ACM, 2010, pp. 385–392.
[125] T. Suttorp, N. Hansen, and C. Igel, "Efficient covariance matrix update for variable metric evolution strategies," *MLJ*, vol. 75, pp. 167–197, 2009.
[126] C. Igel, T. Suttorp, and N. Hansen, "A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies," in *GECCO*. Seattle, Washington, USA: ACM, 2006, pp. 453–460.
[127] H. Wang, M. Emmerich, and T. Bäck, "Mirrored orthogonal sampling for covariance matrix adaptation evolution strategies," *ECJ*, vol. 27, no. 4, pp. 699–725, 2019.
[128] D. Czégel, I. Zachar, and E. Szathmáry, "Multilevel selection as bayesian inference, major transitions in individuality as structure learning," *Royal Society Open Science*, vol. 6, no. 8, p. 190202, 2019.
[129] E. Szathmáry and J. M. Smith, "The major evolutionary transitions," *Nature*, vol. 374, pp. 227–232, 1995.
[130] S. Okasha, "Multilevel selection and the major transitions in evolution," *Philosophy of Science*, vol. 72, no. 5, pp. 1013–1025, 2005.
[131] G. O. Bozdag, S. A. Zamani-Dahaj *et al.*, "De novo evolution of macroscopic multicellularity," *Nature*, vol. 617, pp. 747–754, 2023.
[132] E. Szathmáry, "To group or not to group?" *Science*, vol. 334, no. 6063, pp. 1648–1649, 2011.
[133] M. J. Flynn, "Very high-speed computing systems," *PIEEE*, vol. 54, no. 12, pp. 1901–1909, 1966.
[134] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms," *TEVC*, vol. 6, no. 5, pp. 443–462, 2002.
[135] M. Thamm and B. Rosenow, "Machine learning optimization of majorana hybrid nanowires," *PRL*, vol. 130, no. 11, p. 116202, 2023.
[136] A. C. Y. Li, A. Macridin, S. Mrenna, and P. Spentzouris, "Simulating scalar field theories on quantum computers with limited resources," *PRA*, vol. 107, no. 3, p. 032603, 2023.
[137] X. Bonet-Monroig, H. Wang, and et al., "Performance comparison of optimization methods on variational quantum algorithms," *PRA*, vol. 107, no. 3, p. 032407, 2023.
[138] C. Igel, N. Hansen, and S. Roth, "Covariance matrix adaptation for multi-objective optimization," *ECJ*, vol. 15, no. 1, pp. 1–28, 2007.
[139] K. Bringmann, T. Friedrich, C. Igel, and T. Voß, "Speeding up many-objective optimization by monte carlo approximations," *AIJ*, vol. 204, pp. 22–29, 2013.
[140] O. M. Shir, M. Emmerich, and T. Bäck, "Adaptive niche radii and niche shapes approaches for niching with the CMA-ES," *ECJ*, vol. 18, no. 1, pp. 97–126, 2010.
[141] R. Lohmann, "Application of evolution strategy in parallel populations," in *PPSN*, H.-P. Schwefel and R. Männer, Eds., vol. 496. Dortmund, Germany: Springer, 1990, pp. 198–208.
[142] A. H. G. R. Kan and G. T. Timmer, "Stochastic global optimization methods part ii: Multi level methods," *MP*, vol. 39, no. 1, pp. 57–78, 1987.
[143] P. Moritz, R. Nishihara *et al.*, "Ray: A distributed framework for emerging AI applications," in *OSDI*. Carlsbad, CA, USA: USENIX Association, 2018, pp. 561–577.
[144] J. Wakunda and A. Zell, "Median-selection for parallel steady-state evolution strategies," in *PPSN*, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds., vol. 1917. Paris, France: Springer Berlin Heidelberg, 2000, pp. 405–414.
[145] M. G. Arenas, P. Collet, and et al., "A framework for distributed evolutionary algorithms," in *PPSN*, J. J. M. Guervós, P. Adamidis, H.-G. Beyer, H.-P. Schwefel, and J.-L. Fernández-Villacañas, Eds., vol. 2439. Granada, Spain: Springer Berlin Heidelberg, 2002, pp. 665–675.
[146] M. Biazzini, "Gossiping optimization framework (GOOF): A decentralized p2p architecture for function optimization," Ph.D. dissertation, University of Trento, 2010.
[147] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *CACM*, vol. 51, no. 1, pp. 107–113, 2008.
[148] X. Meng, J. Bradley *et al.*, "MLlib: Machine learning in apache spark," *JMLR*, vol. 17, no. 34, pp. 1–7, 2016.
[149] F. Bizzaro, M. Conti, and M. S. Pini, "Proof of evolution: Leveraging blockchain mining for a cooperative execution of genetic algorithms," in *IEEE International Conference on Blockchain*, Rhodes, Greece, 2020, pp. 450–455.
[150] J. Clune, J. Lehman, and K. Stanley, "Recent advances in population-based search: Quality diversity, open-ended algorithms, and indirect encodings," 2019.
[151] M. Jaderberg, V. Dalibard *et al.*, "Population based training of neural networks," 2017.
[152] E. Real, C. Liang, D. So, and Q. Le, "Automl-zero: Evolving machine learning algorithms from scratch," in *ICML*, H. D. III and A. Singh, Eds., vol. 119. Virtual: PMLR, 2020, pp. 8007–8019.
[153] R. Wang, J. Lehman *et al.*, "Enhanced POET: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions," in *ICML*, H. D. III and A. Singh, Eds., vol. 119. Virtual: PMLR, 2020, pp. 9940–9951.
[154] M. López-Ibáñez, J. Branke, and L. Paquete, "Reproducibility in evolutionary computation," *ACM-TELO*, vol. 1, no. 4, pp. 14:1–21, 2021.
[155] D. Whitley, S. Rana, J. Dzubera, and K. E. Mathias, "Evaluating evolutionary algorithms," *AIJ*, vol. 85, no. 1, pp. 245–276, 1996.
[156] P. Bennet, C. Doerr, A. Moreau, J. Rapin, F. Teytaud, and O. Teytaud, "Nevergrad: Black-box optimization platform," *SIGEVOlution*, vol. 14, no. 1, pp. 8–15, 2021.
[157] I. L. Markov, "Limits on fundamental limits to computation," *Nature*, vol. 512, pp. 147–154, 2014.
[158] J. P. Cohoon, S. U. Hegde, W. N. Martin, and D. Richards, "Punctuated equilibria: A parallel genetic algorithm," in *ICGA*. Cambridge Massachusetts USA: L. Erlbaum Associates Inc., 1987, pp. 148–154.
[159] N. Hansen and S. Kern, "Evaluating the CMA evolution strategy on multimodal test functions," in *PPSN*, X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. E. Rowe, P. Tiňo,

A. Kabán, and H.-P. Schwefel, Eds., vol. 3242. Birmingham, UK: Springer Berlin Heidelberg, 2004, pp. 282–291.

[160] P. Larrañaga, Ed., *Estimation of distribution algorithms: A new tool for evolutionary computation*. Springer, 2002.

[161] Q. Duan, G. Zhou, C. Shao, Y. Yang, and Y. Shi, "Collective learning of low-memory matrix adaptation for large-scale black-box optimization," in *PPSN*, G. Rudolph, A. V. Kononova, H. Aguirre, P. Kerschke, G. Ochoa, and T. Tušar, Eds., vol. 13399. Springer International Publishing, 2022, pp. 281–294.

[162] C. Aranha, C. L. Camacho Villalón, F. Campelo, M. Dorigo, R. Ruiz, M. Sevaux, K. Sörensen, and T. Stützle, "Metaphor-based metaheuristics, a call for action: The elephant in the room," *Swarm Intelligence*, vol. 16, no. 1, pp. 1–6, 2022.