Elsevier required licence: ©2024. This manuscript version is made available under the CCBY-NC-ND 4.0 license http://creativecommons.org/licenses/by-nc-nd/4.0/ The definitive publisher version is available online at https://doi.org/10.1016/j.asoc.2024.112232

Cooperative Coevolution for Non-Separable Large-Scale Black-Box Optimization: Convergence Analyses and Distributed Accelerations

Qiqi Duan, Chang Shao, Guochen Zhou, Haobin Yang, Qi Zhao, and Yuhui Shi, *Fellow, IEEE*

***Abstract*—Given the ubiquity of non-separable optimization problems in real worlds, in this paper we analyze and extend the large-scale version of the well-known cooperative coevolution (CC), a divide-and-conquer optimization framework, on non-separable functions. First, we reveal empirical reasons of why decomposition-based methods are preferred or not in practice on some non-separable large-scale problems, which have not been clearly pointed out in many previous CC papers. Then, we formalize CC to a continuous game model via simplification, but without losing its essential property. Different from previous evolutionary game theory for CC, our new model provides a much simpler but useful viewpoint to analyze its convergence, since only the pure Nash equilibrium concept is needed and more general fitness landscapes can be explicitly considered. Based on convergence analyses, we propose a hierarchical decomposition strategy for better generalization, as for any decomposition there is a risk of getting trapped into a suboptimal Nash equilibrium. Finally, we use powerful distributed computing to accelerate it under the multi-level learning framework, which combines the fine-tuning ability from decomposition with the invariance property of CMA-ES. Experiments on a set of high-dimensional functions validate both its search performance and scalability (w.r.t. CPU cores) on a clustering computing platform with 400 CPU cores.**

***Index Terms*—Black-box optimization, convergence, cooperative coevolution, distributed algorithm, large-scale optimization.**

I. INTRODUCTION

RECENT advances in artificial intelligence (particularly deep models [1], [2], [3]) and big data have generated a growing number of large-scale optimization (LSO) problems, including challenging high-dimensional black-box optimization (BBO) instances [4], [5]. An *OpenAI* team [6] used a highly-parallelized version of evolution strategies (ES) to optimize millions of weights of deep neural networks [7] (DNN) for direct policy search in reinforcement learning (RL). In a recent *Science* paper, Fan *et al.* [8] hybridized genetic algorithm (GA) and simulated annealing (SA) to solve a big-data-based paleobiology model on a Chinese high-performing supercomputer. A *DeepMind* team proposed a population-based training method for hyperparameter optimization (HPO) of generative adversarial networks [9], multi-agent RL [10], and deep vision model for self-driving cars [11].

Although evolutionary algorithms (EAs [12], [13], [14]) are a very popular algorithm family for BBO, nearly all standard versions of EAs have to be significantly improved in the LSO context, since they suffer easily from the notorious *curse of dimensionality*. For a survey of EAs for LSO, see e.g., [15], [16], [17] and references wherein. For efficient search in large-scale space, exploiting the possibly useful problem structure is a critical step to accelerate convergence progress, as shown in Nesterov's outstanding optimization book [18]. In this paper, we focus on mainly two of representative EAs both with some successful LSO applications: 1) cooperative co-evolution (CC [19]-[24]) based on the (co-adapted) modularity assumption; and 2) covariance matrix adaptation evolution strategies (CMA-ES [25], [26]), which is regarded widely as the state-of-the-art for BBO in a recent *Nature* review for EAs [13]. For modern CC, CMA-ES is often chosen as the basic suboptimizer for all subproblems, because of its well-studied theoretical properties (invariance against affine transformation [27], unbiases under neutral selection [28], maximum entropy (diversity) principle [29], learning of natural gradient [30], [31]) and generalizable search abilities (in particular on a family of *non-separable* and *ill-conditioned* problems).

In 2014, a modern CC variant with the so-called differential grouping (DG) technique [32] obtained promising results on a class of *partially additively separable* (PAS) [33] benchmark functions for LSO (mainly from IEEE-CEC competitions [34], [35], [36]). Its well-established theoretical foundation to detect variable interactions has now sparked many following-up works (see Section II.A). However, as previously argued in a *Nature* review [37], "*most instances of most problems are not readily 'linearly' decomposable into building blocks*". Indeed, a plenty of real-world problems have complex (*nonlinear*) objective functions, where typically there have *explicit* or *implicit* interactions between any two variables for both gradient-based optimization [38] and BBO. Take e.g., [39]-[49] as examples, to name a few[1]. Even linear regression,

This work was supported in part by Shenzhen Fundamental Research Program under Grant No. JCYJ20200109141235597 and Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant No. 2017ZT07X386. *(Corresponding author: Yuhui Shi).* Qiqi Duan and Chang Shao contributed equally.

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 and Southern University of Science and Technology, Shenzhen, China.

Haobin Yang, 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).

[1] Here we provide an online website (https://tinyurl.com/3d4nnneb) which covers many real-world applications (nearly all of them are *non-separable*). *What kinds of real-world applications are PAS* is still an open question.

perhaps the simplest data model, has a *fully nonseparable* loss function form[2] [50] for least-squares estimation. In fact, in the original Ph.D. dissertation regarding CC [20], all of three real-world problems considered (i.e., string cover, rule learning, and neuroevolution) are *non-separable*, though CC was first benchmarked on both *separable* and *nonseparable* artificially-constructed functions. Currently its state-of-the-art real-world applications come mainly from neuroevolution for RL (refer to Miikkulainen's or Schmidhuber's lab [51]-[58]). Obviously, all the loss functions used by them are *non-separable*, caused by *nonlinearity* of NN itself as well as the simulation model.

Given the ubiquity of non-separable problems in practice, in this paper we focus on CC for non-separable large-scale BBO mainly from three different yet related viewpoints (problem structure, convergence analyses, and distributed acceleration).

Problem Structure. It is natural to deduce that not all non-separable problems can be handled efficiently by CC. For example, on many *ill-conditioned* non-separable landscapes, both CC and its gradient-based counterpart (i.e., coordinate descent, CD [59]) are typically *worse* than second-order-type optimizers (e.g., LM-CMA [60] and L-BFGS [61]). On the contrary, on some other non-separable landscapes (e.g., with relatively *sparse* variable interactions [62]), some modern CC could obtain very competitive (sometimes even state-of-the-art) results. Therefore, a key theoretical question arises: *On what kinds of problem types from (non-separable) real-world problems CC is preferred over others?* As argued in 1989 by Conn et al. [63], PAS functions "*are clearly a very restricted case of partially separable (PS) functions*"[3]. Here we exclude it because of its hard-to-satisfy assumption in practice.

To the best of our knowledge, there have no theoretical work to satisfactorily solve the above theoretical challenge. In this paper, we first answer a related but much simpler practical question: *When may end-users prefer to use divide-and-conquer methods in practice?* Some potential answers are presented in the following:

- There often exists a "*natural*" decomposition for many complex systems: e.g., body-brain co-evolution [39], [64] and multi-agent learning [65] in the evolutionary robotics field, and expectation-maximization [66], [67] from the AI field. Their significant feature is that all the subcomponents interact *nonlinearly* but work *at different time scales or computing units* (leading to different update frequencies). Note that the so-called 'natural' decomposition reflects the design preference (*not necessarily* the 'optimal' decomposition solution).
- In parallel/distributed computing, a single computing unit (e.g., a CPU core) always has a relatively limited memory and capability with Moore Law's ending [68]. For big-data driven LSO with certain structures [69], [70], [71], general decomposition strategies (e.g., CD) are a viable solution for scalability, one critical metric for any general-purpose optimizer.
- There may be a relatively *weak* interaction between subcomponents in some (not all) non-separable real-world optimization problems, where decomposition-based methods could converge fast (with practically accepted accuracy). This may partly explain why there *always* have some researchers and practitioners to use them since the establishment of the optimization area (furthermore, another advantage is their relative ease to understand and implement) [59], [71].

Convergence Analyses. In order to make it mathematically tractable, we simplify CC as a continuous game (CG) model [72] but without losing its essential property. As compared to previous evolutionary game theory (EGT) [65], [73], our CG model can provide a *much clearer* analytical perspective built on only the pure Nash equilibrium (PNE), simpler than its mixed counterparts used in EGT. We theoretically show that *under what conditions* CC convergences, which most convex-quadratic benchmark functions can satisfy (see Section III). To validate the prediction ability, we further demonstrate that many *overlapping* functions from the two newest test suites could *essentially* satisfy these conditions, which illustrates that CC finally converges to the global optimum on them, which is proved for the first time (our experiments can *perfectly* match our predictions). Furthermore, using this new model, we re-confirm previously discovered pathologies of CC in a unified manner (i.e., relative generalization [74] and loss of gradients [75]). Depending upon these theoretical analyses, we propose to use a *hierarchical decomposition* structure to alleviate these issues and to obtain better generalizability (see Section IV.A).

Distributed Acceleration. Intuitively, CC appears to fit for distributed computing well [19]. However, under nonlinear interdependencies between subcomponents, the parallelism of CC is a non-trivial task, since such a nonlinearity can lead to a difficulty in credit assignment for fitness evaluations of each subpopulation. To bypass this, we use the recently proposed *multi-level learning/evolution* (MLE) framework [76], [77], where each subpopulation conducts (local) metric learning on only its corresponding subspace for a much lower time and space complexity, which is important to match the hierarchical memory structure of CPU. At the same time, multiple large-scale CMA-ES variants are maintained in parallel, in order to reduce the (possible) risk of getting trapped into a suboptimal Nash equilibrium encountered by CC. After each relatively short learning period, all learnt information will be collected, selected, and diversified at the meta-level for next cycles. Overall, the MLE framework can combine the fine-tuning ability from decomposition with the powerful invariance [78] property of CMA-ES (see Section IV).

Experiments on a set of high-dimensional functions validate its search performance and scalability on an industry-level clustering computing platform with 400 cores (see Section V).

[2] Only when the involved data has a block matrix structure, the resulting objective function is PAS. To our knowledge, however, little of real-world applications exhibit this (see https://archive.ics.uci.edu/ml/datasets.php for > 150 data set). In general, its objective function is *fully nonseparable* (here we do not consider its complex *dual* form, which is clearly out of scope).

[3] We notice that there are several CC papers do not distinguish PAS and PS clearly, which may cause confusions. In the mathematical optimization (MO) community, PS includes not only *PAS*, but also *non-separable functions with a sparse linkage structure*. In effect, the focus of MO is the latter rather than the former, as the former is seen as "*a very restricted case*", which is not emphasized by some CC papers.

II. Related Works

In this section, we first analyze the state-of-the-art CC and then discuss its real-world applications with non-separable forms and related theoretical advances. Next, we check many of *partially separable* real-world problems from the traditional mathematical optimization community, to show that nearly all of them are *non-separable* according to variable interactions. Finally, following the suggestion from the latest CC survey [24], we build a connection between CC and its gradient-based counterpart (i.e., coordinate descent) to better understand the essence of decomposition-based optimizers.

A. State-of-the-Art CC

Since [32], a series of different improvements based mainly on DG (e.g., [82]-[98]) have been proposed. Although these improvements often reduce the needed number of function evaluations, they might become over-skilled because they are built on the PAS assumption[4]. As stated before, *what kinds of real-world applications satisfy the PAS assumption* is still an open question. Similar issue has happened in GA's building-block hypothesis (BBH) for crossover operators (refer to [29], [99]). Like BBH, while PAS is intuitively appealing, finding functions from real-world applications to support it appeared surprisingly difficult. Although these automatic separability detection techniques sometimes could be used to analyze the variables interaction matrix as the basis of possible problem transformation for gray-box optimization [22], such a problem transformation is unavailable in box-box scenarios. Note that the real-world problem itself in [22] is *non-separable* (after transformation its new form is still *non-separable*). Similarly, in both [89] and [98], their tested real-world problem is also *non-separable*, despite they focused on decomposition.

Till now, decomposition of *non-separable* problems is still a challenge. The above separability detection will become of less importance for non-separable functions when such *a prior* knowledge (i.e., *non-separability*) is relatively easy to obtain in practice (e.g., [100], [51] to name but a few). Recently, Chen et al. [101] considered *non-additively partially separable* problems [102]. However, how to extend it for general *non-separable* problems is still unclear. Komarnicki et al. [103] suffered from the same issue. Zhang et al. [104] applied CC to *non-separable* problems and obtained promising results on some benchmark functions. However, they did not analyze its convergence property. Jia et al. [105] extended contribution-based CC to a special type of *non-separable* functions (called *overlapping* [62], [106], [36]). Similarly, they did not analyze whether CC converges or not yet. Although recently Ge et al. [86] presented a simple convergence analysis under the block separability assumption, it is not suitable for *non-separability*.

B. Theoretical Advances of CC

There have been a series of theoretical works based on EGT to analyze complex convergence behaviors of CC on non-separable functions. Wiegand [74] for the first time proved that its replicator equations converge to PNE and showed that CC sometimes suffers from the *relative generalization* issue. Although the following-up works (e.g., [73], [107], [108]) extended Wiegand's work somewhat, they are not extended to current large-scale CC versions due to the following factors: 1) It considers only one very simple decomposition case on a (discretized) low-dimensional search space; 2) it depends on a highly simplified payoff matrix for computing mixed Nash equilibria and therefore it cannot consider complex fitness landscape explicitly; 3) its derived lenient learning is still rarely used for LSO [65]. Overall, although they could provide a valuable analytical tool for CC, new theoretical advances are still expected for LSO-focused CC.

In [109] and [110], Jansen and Wiegand provided the first expected runtime analysis for CC. Surprisingly, they found that "*the property of separability is neither a sufficient one to imply an advantage of CC, nor is inseparability a sufficient enough property to imply a disadvantage*". Similarly, Gomez et al. [51] argued that "*much of the motivation for using the CC approach is based on the intuition that many problems may be decomposable into weakly coupled low-dimensional subspaces that can be searched semi-independently by separate species. Our experience shows that there may be another, complementary, explanation as to why cooperative coevolution in many cases outperforms single-population algorithms*".

C. Partially Separability and Coordinate Descent

Originally, the concept of partially separability (PS) [111] was defined in 1981 from the mathematical optimization field and is still studied by its one original author Toint now [112]. Until 2010, its very special form (i.e., PAS) was popularized for CC in the large-scale BBO field via a series of IEEE-LSGO competitions, despite there was one earlier (1999) EA paper also involving it [33]. Here, we re-check PS based on the original paper[5]. Surprisingly, there is **only one** function in [111] that meets the PAS assumption in all 43 functions (here we have excluded 7 extra low-dimensional (<5) functions, because they are only used to test programming correctness). Moré et al. [113] collected a total of 24 optimization problems from many real-world applications. **All[6] of** them have a *non-separable* form [114]. Arguably, most PS instances from the mathematical optimization field are *non-separable*, obviously different from previous IEEE-LSGO competitions. Similarly, coordinate descent (CD) is commonly used to optimize non-separable problems (often with friendly structures), since its first application in 1954 [115]. Note that for CD separability is limited to only the regularization term (in other words, the whole function form is generally inseparable).

[4] The PAS assumption may be of a *theoretical* interest for researchers.

[5] The first (1981) paper of partially separability is not accessible online. Unfortunately, the original author (Toint) cannot provide its pdf version (via email communication). Instead, we use his 1983 version, which provides a set of 50 functions. Again, this 1983 paper also becomes inaccessible online now (to our knowledge). Luckily, we have saved it locally before (if you want to read it, please contact Toint or us only for academic purpose).

[6] We have excluded 4 problems since their objective function formula are not given *explicitly*.

III. Continuous Games and Convergence

In this section, we propose a continuous game model as a base of convergence analyses of CC. To validate its prediction ability, the convergence behaviors of CC are analyzed on common fitness models and the latest *overlapping* functions.

A. Continuous Games (CG) Model of CC

For any objective function[7] $f(\boldsymbol{x})\colon \mathbb{R}^n \to \mathbb{R}$, CC divides all its coordinate indexes $\{1,\dots,n\}$ into a set of *mutually exclusive* partitioning groups $p = \{g_1,\dots,g_m\}$ where $1 < m \le n$ and $g_i \ne \emptyset$, $g_i \cap g_{j\ne i} = \emptyset$, $\cup_i g_i = \{1,\dots,n\}$, $\forall i,j \in \{1,\dots,m\}$. Note that $1 < m$ excludes no decomposition. An interesting theoretical question is *how many* possible partitions exist for decomposition-based methods. The Bell number, a computing concept from **combinatorial mathematics**, can help answer this question. For example, even for a 25-d function, there are totally 4,638,590,332,229,999,352 partitioning solutions (even when the partitioning order is not considered) [116].

1. *Ubiquity of Pure Nash Equilibrium in CG*

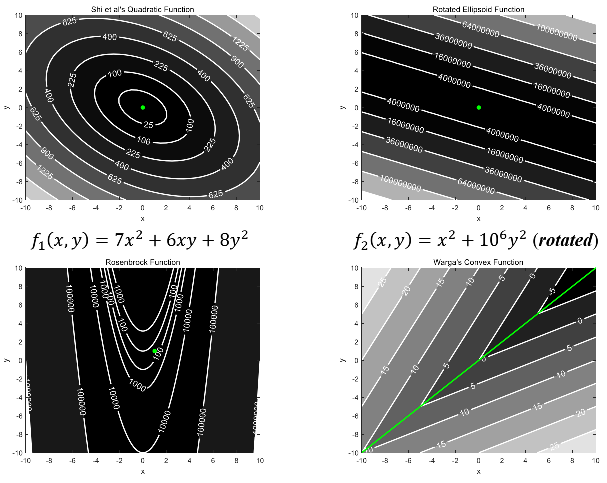
In game theory, it is well-known that there *always* exists at least one *mixed* (not necessarily *pure*) Nash equilibrium for any noncooperative game [117], [118]. However, our previous paper has mathematically shown that CG's special payoff structure guarantees the ubiquity of the pure Nash equilibrium (PNE) for *any* one partitioning (refer to [119] for details). Hereinafter, in order to avoid ambiguity, we will focus on only the PNE as defined below.

Definition (**Pure Nash Equilibrium**). Given *any* partition $p = \{g_1,\dots,g_m\}$ of the fitness function $f(\boldsymbol{x})\colon \mathbb{R}^n \to \mathbb{R}$, we say that its decision vector $\boldsymbol{x} \in \mathbb{R}^n$ is one *pure Nash equilibrium* (w.r.t. p) iif the decision subvector $\boldsymbol{x}_{g_i} \in \mathbb{R}^{|g_i|}$ for each $g_i \in p, \forall i \in \{1,\dots,m\}$, satisfies

$$f(\boldsymbol{x}_{g_i}, \boldsymbol{x}_{\ne g_i}) \le f(\boldsymbol{x}_{g_i}^{\sim}, \boldsymbol{x}_{\ne g_i}),\ \forall \boldsymbol{x}_{g_i}^{\sim} \in \mathbb{R}^{|g_i|} \backslash \{\boldsymbol{x}_{g_i}\}, \qquad (1)$$

where $\boldsymbol{x}_{\ne g_i}$ denotes all the remaining decision subvectors excluding $\boldsymbol{x}_{g_i}$ and $\boldsymbol{x}_{g_i}^{\sim}$ denotes any other possible values (i.e., $\mathbb{R}^{|g_i|} \backslash \{\boldsymbol{x}_{g_i}\}$). If (1) is strict, we say that $\boldsymbol{x}$ is a *strict* PNE (w.r.t. p). Since this concept relies heavily on the pre-partition, we add a suffix "(w.r.t. p)" if necessary. To help understand this solution concept for decomposition-based optimization, we visualize some of its common distributions on four *non-separable* functions in Fig. 1.

As is seen in Fig. 1, the top-left is a *strictly-convex* function, designed by Shi et al. [71] from the latest CD survey. There is a unique PNE, which is the global optimum (CC could converge to it rapidly owing to weak dependency). The top-right is an *ill-conditioned* (rotated) Ellipsoid function, used to benchmark local search ability of EAs. Similarly, there is only one PNE (CC still could converge to it but the convergence rate is very slow owing to strong dependency). The bottom-left is a very famous *non-convex* function [120], where there is also a unique PNE (CC still could converge to it but the convergence rate is slow when approaching the parabolic curve). The bottom-right is a *convex but non-differentiable*



$f_1(x,y) = 7x^2 + 6xy + 8y^2$ $\quad$ $f_2(x,y) = x^2 + 10^6 y^2$ ***(rotated)***

$f_3(x,y) = 100(x^2 - y)^2 + (x-1)^2$ $\quad$ $f_4(x,y) = |x-y| - min\,(x,y)$

Fig. 1. Visualization of all pure Nash equilibria on four 2-d non-separable functions. The white line represents the contour levels of fitness landscape while the green point/line (best-response curve) denotes its location.

function, originally proposed by Warga [121] from the CD community. For it, there is *a continuum of* PNE (CC does not necessarily converge to the global optimum but to other suboptimal PNE, depending on the starting point). For their strict convergence demonstration, see online Supplementary Materials[8].

Lemma 1: Given *any* partition $p = \{g_1,\dots,g_m\}$ of $f(\boldsymbol{x})$, its global optimum $\boldsymbol{x}^*$ is a pure Nash equilibrium w.r.t. p. However, the inverse is *not necessarily* true.

The ubiquity of PNE under *any* partition can be guaranteed by the above lemma, only if the objective function has (at least) one global optimum (often default).

Lemma 2: Given *any* partition $p = \{g_1,\dots,g_m\}$ of $f(\boldsymbol{x})\colon \mathbb{R}^{n>2} \to \mathbb{R}$ where $|g_i| > 1, \exists i \in \{1,\dots,m\}$, we can divide p along g_i and get a new partition p' satisfying $|p'| > |p|$. If $\boldsymbol{x}$ is a PNE w.r.t. p, then $\boldsymbol{x}$ is also a PNE w.r.t. p'. However, the inverse does not necessarily hold. (We term this interesting property as ***downward rather upward propagation of PNE***.)

Since *lemma 2* plays a fundamental role in the following convergence analyses, we design an example to illustrate it. Consider an artificially designed function $f(x,y,z) = f(x,y) + f(y,z)$, where $f(x,y) = (x+y)^2 + (y-1)^2$ and $f(y,z) = (y+1)^2 + (y+z)^2$. For it, there are 4 partitions: $\{\{1,2\},\{3\}\}$, $\{\{1,3\},\{2\}\}$, $\{\{2,3\},\{1\}\}$, and $\{\{1\},\{2\},\{3\}\}$, as shown in Fig. 2 (for simplicity, the order of partitions is not considered here). Interestingly, all partitions can be well re-organized in a *hierarchical* (*recursive*) manner. Each green arrow represents one further partition operation (i.e., from p to p'). The global optima, a special type of PNE, can propagate both *downwardly* (from p to p') and *upwardly* (from p' to p). One theoretical significance of *lemma 2* is that it can greatly simplify the seeking of PNE, as proved in the next section.

2. *Convergence to a PNE in CG*

Although it is well-known in the scientific community that "*essentially all models are wrong*" [73], a good model for CC should meet (at least) the following two criteria while omitting

[7] Only the *minimization* of objective (fitness) functions is considered, since maximization can be transformed into minimization simply by negating it.

[8] https://github.com/Evolutionary-Intelligence/DCC

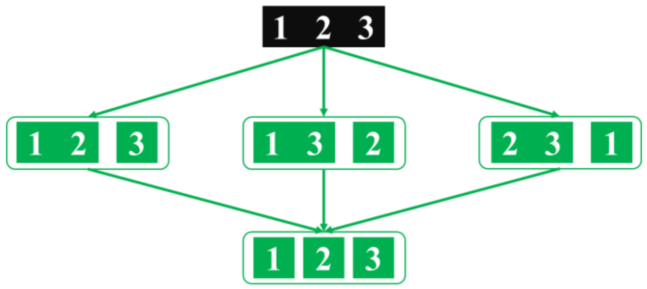


Fig. 2. A simple example to illustrate the *downward rather upward propagation* of PNE along hierarchical partitions.

the peripheral details: 1) It should still capture the essence of the problem after simplifications, which can be used to predict the *limit* convergence behavior; 2) It should exhibit somewhat *extrapolation* ability on some unseen scenarios, despite it is nearly impossible to exclude the failure risk in practice.

In previous EGT models [73], [74], CC was modeled using two-player replicator equations. Alternatively, CC can also be simply modeled as a continuous game (CG) on the original continuous search space, which is comprised of multiple dynamically-coupled subpopulations. Continuous games have been investigated in many research areas (economics [122], nonlinear automatic control [123], [72], multi-agent learning [124], engineering design [125], and AI [126], [127], etc.).

In principle, our CG model can be seen as a *particular* form of CG proposed by Ratliff et al. [72] from the automatic control community. For the former, all the players have the same objective function form but with a disjoint subset of decision vectors. However, for the latter, each player has its own objective function, which generally has a different form with each other. We borrow the classical model [128], [129] from CD to formalize each cycle of CC, mathematically represented as one ***transformation*** $\boldsymbol{T}: \Omega \subset \mathbb{R}^n \to \Omega \subset \mathbb{R}^n$ of the search space Ω into Ω itself, satisfying two conditions:

1. $f(T(\boldsymbol{x})) \leq f(\boldsymbol{x})$, where $\boldsymbol{x}$ is the *best-so-far* solution;
2. $f(T(\boldsymbol{x})) = f(\boldsymbol{x}) \Leftrightarrow T(\boldsymbol{x}) = \boldsymbol{x}$. (a fixed point)

Condition 2 excludes one very special case (that is, when $f(T(\boldsymbol{x})) = f(\boldsymbol{x}), \boldsymbol{T}(\boldsymbol{x}) \neq \boldsymbol{x}$), which otherwise results in the *cyclical* behavior of the best-so-far solution and prevents the convergence (found first by Powell in 1973 [130]).

For successive transformation processes, the key point is to extract a *strictly decreasing* fitness subsequence from the non-increasing fitness sequence generated by CC. This could be guaranteed by a strong assumption that each suboptimizer has *sufficient* search ability for each subproblem (that is, *it can find the global minimizer for each subproblem given a finite number of iterations*). It is worthwhile noting that even such a strong assumption cannot ensure the convergence to the global optimum on the original problem (see Fig. 1(4)). As a result, finally $\lim_{t\to+\infty} T_t(\boldsymbol{x})$ will move towards **a fixed point** (that is, $\boldsymbol{T}(\boldsymbol{x}) = \boldsymbol{x}$), which is also a PNE[9]. In practice, the global minimizer of each subproblem in each cycle is not needed to find exactly [62].

We acknowledge that the above CG model is very simple from a practical view of point. Based on it, however, we will derive some interesting theoretical results, as presented below.

B. Convergence Analyses on Convex Functions

Based on the proposed CG model, the convergence problem of CC can be well converted to the *seeking* and *classification* problem of PNE. Here we provide a main theorem on convex functions, as presented below:

Main Theorem: Consider *any* partition $p = \{g_1, \dots, g_m\}$ of a continuous-differentiable convex function $f(\boldsymbol{x}) \in C^1(\Omega, \mathbb{R})$, where Ω is an open convex set[10] in $\mathbb{R}^n$. For $f(\boldsymbol{x})$, all PNE are equivalent to global optima, and vice versa.

Proof: First we demonstrate one special partition case $p^n = \{\{1\}, \dots, \{n\}\}$ (that is, $m = n$). Assume $\breve{\boldsymbol{x}}$ is a PNE w.r.t. p^n. By the definition of PNE, we can simply infer that $\breve{\boldsymbol{x}}_{\{i\}} \in \underset{\boldsymbol{x}_{\{i\}}}{argmin}\, f(\boldsymbol{x}_{\{i\}}, \boldsymbol{x}_{\neq\{i\}}) \subseteq \Omega, \forall i \in \{1, \dots, n\}$.

Owing to the continuous differentiability of $f(\boldsymbol{x})$, each partial derivative $\nabla_{\{i\}} f(\breve{\boldsymbol{x}}) = 0, \forall i \in \{1, \dots, n\}$. So, $\breve{\boldsymbol{x}}$ is also a stationary point. Note that for $\underset{\boldsymbol{x}_{\{i\}}}{argmin}\, f(\boldsymbol{x}_{\{i\}}, \boldsymbol{x}_{\neq\{i\}})$, the solution is not necessarily unique. According to the convexity of $f(\boldsymbol{x})$, it is well-known that all stationary points are global optima. With *lemma 1* (that is, all global optima are PNE), we conclude that for $p^n = \{\{1\}, \dots, \{n\}\}$, all PNE w.r.t. p^n are global optima, and vice versa.

Then, we show the general case for any other partition $p = \{g_1, \dots, g_m\}$ when $m \neq n$. Based on *downward propagation* of PNE (*lemma 2*), all PNE w.r.t. p belongs to a set of PNE w.r.t. p^n, which are also global optima. With *lemma 1*, we finish the proof perfectly. □

The above main theorem is valuable to understand why CC could converge to the global optima on many (not all) *non-separable* functions. In effect, many of common benchmark functions fall into this class [131]. Take e.g., *Cigar*, *Discus*, *CigarDiscus*, *Ellipsoid*, and *Schwefel* as examples for convex-quadratic functions [5]. Note that their *non-separable* rotated-and-shifted versions still fall into this class. Surprisingly, all the overlapping functions from the newest LSO test suite for CC [62] essentially still meet these above assumptions (after suitable simplifications). The above main theorem can in part explain why one state-of-the-art CC version could obtain very competitive performance on a particular class of overlapping functions, despite all of them are non-separable (refer to the following subsection for more details). In most of previous CC for LSO, both the theoretical and practical significances of PNE are overlooked or blurred. However, in the non-separable cases, there are more complex relationships between PNE and global optima, which need to be clarified theoretically [132].

C. Convergence Guarantee for some Overlapping Functions

In order to validate the *extrapolation* ability of our proposed CG model, here we consider a class of *overlapping* (a special

[9] Its mathematical proof is simple via proof by contradiction: In fact, the Nobel-prize winner Nash was the first to connect the fixed point with the equilibrium point (now named after him) in his seminar *PNAS* and *Annals of Mathematics* papers, laying the theoretical foundation of modern game theory.

[10] Here it is implicitly assumed that there is (at least) one global optimum in this open convex set.

type of non-separability) functions from the latest LSO test suite [62]. The original experiments reported in [62] showed that on each function the best-so-far solution is far from the global optimum, since a *relatively small* maximum of function evaluations (i.e., 3e6) was used. Based upon corollary 1 shown below, however, we predict that CC could converge to the global optimum on them finally, since they *essentially* meet the assumptions of the main theorem, when two benchmarking operations (i.e., non-smoothing and asymmetry) are removed to make the involved mathematics tractable. To validate our prediction, we increase the maximum of function evaluations from 3e6 to 3e7 significantly. New experiment results are in accordance with our theoretical prediction[11](even when two benchmarking operations are added).

Corollary 1: Consider a non-separable overlapping function $f(\boldsymbol{x}) = \sum_{i=1}^{m} f_i(\boldsymbol{x}_i)$ with $\boldsymbol{x} \in \Omega \subseteq \mathbb{R}^n$, $\boldsymbol{x}_i \cap \boldsymbol{x}_j \neq \emptyset, \exists i \neq j \in \{1, \dots, m\}$ and $\cup_i \boldsymbol{x}_i = \boldsymbol{x}$. All subfunctions f_i have the exactly same [12] base form $f_b(\boldsymbol{x}) = \sum_{i=1}^{n}(\sum_{j=1}^{i} \boldsymbol{x}_i)^2$ (i.e., *Schwefel's Problem 1.2* in [62]). Its rotated-and-shifted version $f_s^r(\boldsymbol{x}) = f_b(\boldsymbol{R}(\boldsymbol{x}-\boldsymbol{s}))$ is used by each $f_i(\boldsymbol{x}_i)$, where $\boldsymbol{R}$ is an orthogonal matrix and $\boldsymbol{s}$ is a shift vector. For this class of overlapping functions, all PNE (w.r.t. *any* partition) are global optima, and vice versa.

Proof: First we show the each continuous-differentiable subfunction $f_i, \forall i \in \{1, \dots, m\}$, is convex. Since all f_i share the same form $f_b(\boldsymbol{x}) = \sum_{i=1}^{n}(\sum_{j=1}^{i} \boldsymbol{x}_j)^2$, we only need to prove the base form case. Its Hessian $\boldsymbol{H}$ is

$$2 * \begin{bmatrix} n & n-1 & n-2 & & 2 & 1 \\ n-1 & n-1 & n-2 & \cdots & 2 & 1 \\ n-2 & n-2 & n-2 & & 2 & 1 \\ & \vdots & & \ddots & & \vdots \\ 2 & 2 & 2 & & 2 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}.$$

Successively adding a multiple of row $i-1$ to another row i by $-\frac{n-(i-1)}{n-(i-2)}$, $i = n, n-1, n-2, \dots, 3, 2$, we get a new matrix:

$$\boldsymbol{A} = 2 * \begin{bmatrix} n & n-1 & n-2 & & 2 & 1 \\ 0 & a_{22} & a_{23} & \cdots & a_{2(n-1)} & a_{2n} \\ 0 & 0 & a_{33} & & a_{3(n-1)} & a_{3n} \\ & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & a_{(n-1)(n-1)} & a_{(n-1)n} \\ 0 & 0 & 0 & & 0 & 1/2 \end{bmatrix},$$

where $a_{ii} = (n-(i-1)) - \frac{n-(i-1)}{n-(i-2)} * (n-(i-1)) > 0, i = n, n-1, n-2, \dots, 3, 2$. By the theorems from linear algebra (pp. 60-61 in [133]), $\boldsymbol{H}$ is row equivalent to $\boldsymbol{A}$. With the theorem from linear algebra (pp. 370 in [133]), we conclude that $\boldsymbol{H}$ is a *positive definite* matrix, which means that $f_b(\boldsymbol{x})$ is a (strictly) convex function.

Second, we consider its rotated-and-shifted variant $f_s^r(\boldsymbol{x}) = f_b(\boldsymbol{R}(\boldsymbol{x}-\boldsymbol{s}))$, where $\boldsymbol{R}(\boldsymbol{x}-\boldsymbol{s})$ is still a convex set. For $\forall \boldsymbol{x} \neq \boldsymbol{y} \in \Omega$ and $\theta \in [0,1]$, $f_s^r(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) = f_b(\boldsymbol{R}(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) - \boldsymbol{s})) = f_b(\theta\boldsymbol{R}(\boldsymbol{x}-\boldsymbol{s}) + (1-\theta)\boldsymbol{R}(\boldsymbol{y}-\boldsymbol{s}))$, by the convexity of f_b, $\leq \theta f_b(\boldsymbol{R}(\boldsymbol{x}-\boldsymbol{s})) + (1-\theta) f_b(\boldsymbol{R}(\boldsymbol{y}-\boldsymbol{s})) = \theta f_s^r(\boldsymbol{x}) + (1-\theta) f_s^r(\boldsymbol{y})$. Therefore, its rotated-and-shifted variant is still a convex function.

Then, if $f_i(\boldsymbol{x}), \forall i = 1 \dots, m$, is a convex function, their non-negative weighted sum is still a convex function (pp. 79 in [134]). A total of 20 overlapping functions in [62], no matter with *conforming* or *conflicting* components, still meet it. With the main theorem, we can conclude the proof. □

Considering that recent CC works focus on the overlapping case, the above corollary can provide a *relatively deep* theoretical understanding to CC's convergence properties on some of overlapping functions with certain structures (the above corollary can also be extended to some overlapping functions in [105], since their base form is also convex).

IV. Distributed Multi-Level Learning

In this section, a distributed cooperative coevolution (DCC) framework is proposed where the recent multilevel learning [77], [76] is employed, in order to improve the efficiency and effectiveness of CC on clustering computing platforms. Given the high complexity of distributed algorithms, we provide the source code available at GitHub[13], to ensure repeatability. For distributed algorithms, multiple performance tradeoffs must be considered: such as, distributed scheduling, distributed (shared) memory management, network communications, fault tolerant, system control, and so on. In this paper, we focus on the *application-level* parallelism and leave other tedious tasks to the underlying distributed computing engine.

A. Hierarchical Structure of DCC

As previously shown, CC tends to convergence to the PNE. If the PNE is not the global optimum (refer to Fig. 1 as an example), CC easily gets trapped into this suboptimal solution. This issue (known as *relative generalization*) is attributed to decomposition, which cannot be avoided in essence but may be alleviated somewhat. On a class of *ill-conditioned* non-separable functions [78], like its gradient-based counterpart (CD), CC often shows *much slower* convergence rates than second-order-type optimizers (e.g., LM-CMA and L-BFGS). However, as a *general-purpose* BBO framework for LSO, it is highly desirable that CC could also handle these challenging issues. To achieve this goal, we use the recently proposed multilevel learning framework [77], [76] to combine the best of both worlds (i.e., the powerful invariance property of LM-CMA and the fine-tuning ability of CC via CMA-ES on low-dimensional subspace). In this paper, we only consider the hierarchical structure to implement it for simplicity, as seen in Fig. 3. In principle, a more general *recursive* structure can be also applied here, which we leave for future works.

As shown in Fig. 3, there are multiple slave nodes as well as one master node for distributed computing (e.g., on Spark [135] and Ray [136]). For DCC, at the meta-level the master node mainly coordinates different optimizers, each of which is run in one separate computing unit (a CPU core). Here we choose two different ES versions (CMA-ES [27] and LM-CMA [60]) as two search engines for the low-dimensional (often <50) subspace (after decomposition) and the original large-scale (>>100) space, respectively. In principle, any other

[11] The source code is freely available at https://bitbucket.org/yuans/rdg3.

[12] In effect, such a condition can be further weakened, only if it is convex and continuous-differentiable. For simplicity, however, we exclude it here.

[13] https://github.com/Evolutionary-Intelligence/DCC

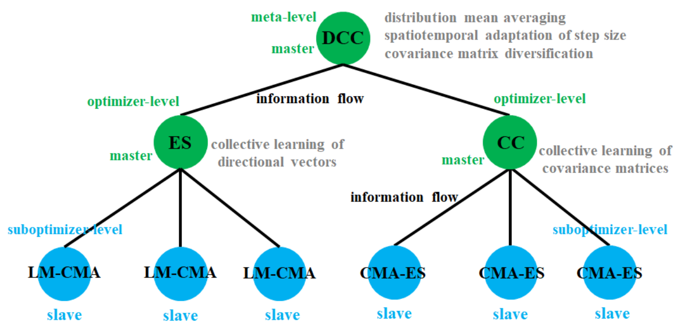


Fig. 3. A hierarchical structure of our distributed cooperative coevolution framework (DCC) for large-scale BBO.

EAs (e.g., GA [137], EP [138], PSO [139], DE [140], and EDA [141]) can be also selected as the suboptimizer. The rationale behind using two search engines lies that in the *original* space LM-CMA has a high possibility to escape from the suboptimal PNE via learning of promising evolution paths while maintaining a low computationally complexity (see [60] for analysis) and in the *decomposed* subspace the full-fledged CMA-ES [27] can well approximate the inverse of its Hessian matrix for fine-tuning with a reasonable memory consumption on each single computing unit (which is vital for scalability of distributed algorithms for LSO). Refer to the following three subsections as well as Algorithm 1 for more details.

B. Collective Learning of Covariance Matrices

For both the LM-CMA and the CMA-ES, covariance matrix adaptation (CMA) plays a central role in their invariance property. Owing to its cubic (or quadratic after modifications) time complexity, CMA is difficult to directly apply to LSO. Instead, approximating CMA with low-memory or low-rank techniques is more acceptable for LSO, resulting in lower time and space complexities (e.g., $O(n * log(n))$ in [60]). For instance, CMA's *rank-one* update (without the *rank-μ* update) can be finally transformed to the following nonrecursive form for offspring sampling (refer to [143] for a detailed deduction):

$$\boldsymbol{d}^t = \left(\left(1-\frac{c_1}{2}\right)\boldsymbol{I} + \frac{c_1}{2}\boldsymbol{p}^1(\boldsymbol{p}^1)^T\right) * \ldots$$
$$* \left(\left(1-\frac{c_1}{2}\right)\boldsymbol{I} + \frac{c_1}{2}\boldsymbol{p}^{t-1}(\boldsymbol{p}^{t-1})^T\right)$$
$$* \left(\left(1-\frac{c_1}{2}\right)\boldsymbol{I} + \frac{c_1}{2}\boldsymbol{p}^{t}(\boldsymbol{p}^{t})^T\right)\boldsymbol{z}^t,$$

where $\boldsymbol{d}^t \in \mathbb{R}^n, \boldsymbol{I} \in \mathbb{R}^{n*n}, \boldsymbol{p}^t \in \mathbb{R}^n, \boldsymbol{z}^t \in \mathbb{R}^n \sim N(\boldsymbol{0}, \boldsymbol{I}), 0 < c_1 < 1$ are the directional vector, identity matrix, evolution path, standard Gaussian permutation, and learning rate of the *rank-one* update for iteration t, respectively. In the actual code implementation, $\boldsymbol{I}$ is omitted since $\boldsymbol{I}\boldsymbol{z}^t = \boldsymbol{z}^t$. The above form should be implemented from right to left, in order to avoid the too expensive matrix operation. Only $m = O(log(n)) \ll n$ dot products are involved for $O(n * log(n))$ complexity.

Although they fit for distributed computing owing to their low memory requirements, these approximations may lose sufficient diversity on the covariance matrix: 1) the *rank-μ* update is not used, which is beneficial for rugged fitness landscape; 2) the *rank-one* update is mainly beneficial for the predominated search direction (e.g., *Cigar* and *Rosenbrock* [5]) rather than multiple promising search directions (e.g., *Discus* and *Ellipsoid* [5]); 3) the *exponentially smoothing* update exploits mainly the *temporal* information but not the (parallel) *spatial* information, which can be alleviated via the distributed

Algorithm 1. Distributed Cooperative Co-evolution (DCC) with CMA-based Multilevel Learning/Evolution. (Refer to the open-source code for repeatability: https://github.com/Evolutionary-Intelligence/DCC.)

```
01: Input: $p$ – total number of available slave nodes in computing platform,
02:     $p\_es$, $p\_cc$– number of slave nodes to run LM-CMA and CC, resp..
03: Output: $b\_x$ , $b\_y$ - best-so-far solution and fitness.
04: Initialize: $b\_x \leftarrow Inf$ , $b\_y \leftarrow Inf$,
05:     randomly initialize LM-CMA and CMA-ES in CC. // in parallel
06: Repeat:
07:     For $i$ from 1 to $p$: // in parallel
08:         If $i \le p\_es$: // for LM-CMA
09:             use Meta-ES to set global step-size,
10:             use elitist or weighted averaging to set distribution mean
11:             run serial LM-CMA in original search space,
12:         Else: // for CMA-ES in $p\_cc$ CC
13:             decompose original space into $k$ subspaces,
14:             For $d$ from 1 to $k$: // in serial for each CC
15:                 run serial CMA-ES in $d$-th subspace // only in low dimensions
16:             End
17:         End
18:     End
19:     synchronize results from all optimizers, // expensive operation
20:     // the following three parts are executed serially:
21:     update distribution mean via weighted averaging at meta-level,
22:     conduct collective learning of covariance matrices, // meta-diversity
23      update $b\_x$ , $b\_y$ if a better is found.
24: Until one termination condition is satisfied.
```

algorithms like Meta-ES [29].

In this subsection, we use the collective learning (or called collective intelligence by Schwefel [142]) paradigm to exploit both the spatial and temporal information for CMA, since this paradigm can naturally match distributed computing. On each slave node, its corresponding ES learns different covariance matrices (directional vectors) periodically. Different form LM-CMA, for all the CMA-ES only the low-dimensional subspace is searched, which needs an extra *assembling* operation in the optimizer-level of CC (see Fig. 3). After each *relatively short* learning period, all information is collected into the meta-level of DCC. Since the temporal information has been exploited on the slave nodes, we average all the covariance matrices of the better LM-CMA instances (e.g., fixed to 1/5 of all instances) as its base of the next generation. The (weighted) averaging does not involve the expensive matrix multiplication operation at the meta-level, which otherwise can severely degrade the speedup on mainstream distributed computing systems based on the master-slave architecture (according to the well-known Amdahl's Law). For better diversity of covariance matrix at the meta-level, we assemble a set of direction vectors from LM-CMA with covariance matrices from CMA-ES for some new LM-CMA instances in the next generation.

Overall, it is expected to maintain the powerful invariance property of LM-CMA especially for ill-conditioned landscape (a challenge for CC), while keeping the fine-tuning ability of CC over a cycle of different subspaces. Our theoretical results have shown that CC tends to converge to the global optimum under certain conditions (even given any decomposition).

C. Distributed Meta-ES for Global Step-Size Adaptation

In the serial CMA-ES [28], the global step-size adaptation is decoupled with the adaptation of covariance matrix, since they can work independently at different time scales for better convergence progress. In the standard CMA-ES form, the cumulative step-size adaptation (CSA) based on the evolution path is used to set the global step-size on-the-fly, which exploits the temporal correlation information over successive generations to speed up convergence significantly. However, in the distributed computing context, the CSA can be further improved, since the spatial information becomes accessible in parallel, resulting in the Meta-ES [29]. Following our previous work [76], we combine Meta-ES with CSA to enjoy the best of both worlds. Specifically, Meta-ES is run at the meta-level of DCC to keep spatial diversity while CSA is employed for each serial ES instance to mainly exploit temporal correlation.

D. Distribution Mean Update via Weighted Averaging

The mean update of the normal search distribution has a significant impact on both the stability and effectiveness of DCC. Following the theoretical work from Beyer [29], we use the (weighted) averaging strategy at the meta-level, in order to obtain the *genetic repair* effect, which is also consistent with all ESs in slave nodes. Note that this simple strategy is also used in some gradient-based distributed optimizers (e.g., for DNN training). However, we have previously observed in [76] that sometimes it could result in the *regression* issue on some functions, which was also identified by Rudolph in [144]. To stabilize the evolution process and keep diversity at the meta-level, we also keep maintaining a relatively small set of *elitist* LM-CMA instances in parallel at each learning period (e.g., fixed to 1/20, typically depending on the number of available computing resources).

In summary, we use a state-of-the-art clustering computing system called **Ray** [136] developed mainly by a team from UC Berkeley to implement our distributed algorithm (DCC).

V. Numerical Experiments

In this section, large-scale numerical experiments on a set of high-dimensional test functions are conducted, in order to show the advantages (and possible disadvantages) of DCC for large-scale black-box optimization.

A. Benchmarking Algorithms

We choose a total of 43 benchmarking baselines, which are classified into the following 10 main families: CC, ES, natural evolution strategies, cross-entropy methods, estimation of distribution algorithms, differential evolution, particle swarm optimization, genetic algorithms, simulated annealing, and random search[14]. For each algorithmic family, there are some standard versions and the latest large-scale variants. All their source code is taken from a recently developed open-source pure-Python library for population-based optimization (called **PyPop7**[15]), which is actively maintained by us now. Owing to page limits, please refer to https://pypop.readthedocs.io for their comprehensive references.

B. High-Dimensional Test Functions

Since the focus of this paper is *non-separable* large-scale BBO, we choose 10 high-dimensional test functions, which are often used to analyze the convergence property in the ES community. For modern ESs, invariance is one fundamental design principle when optimizing non-separable functions. See

TABLE I. A SET OF 10 TEST FUNCTIONS.

Function Name	Mathematic Formulation
$sphere$	$f(x) = \sum_{i=1}^{n} x_i^2$
$cigar$	$f(x) = x_1^2 + 10^6 \sum_{i=2}^{n} x_i^2$
$discus$	$f(x) = 10^6 x_1^2 + \sum_{i=2}^{n} x_i^2$
$cigar\_discus$	$f(x) = x_i^2 + 10^4 \sum_{i=2}^{n-1} x_i^2 + 10^6 x_n^2$
$ellipsoid$	$f(x) = \sum_{i=1}^{n} 10^{6(i-1)/(n-1)} x_i^2$
$different\_powers$	$f(x) = \sum_{i=1}^{n} x_i^{2+4(i-1)/(n-1)}$
$schwefel221$	$f(x) = max(\lvert x_i \rvert)$
$step$	$f(x) = \sum_{i=1}^{n} \left(\sqcup\ (x_i + 0.5)\right)^2$
$rosenbrock$	$f(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$
$schwefel12$	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$

Table I for their detailed formula. As a standard benchmarking [145] practice, we use the *rotation* and *shift* operation to make the test function non-separable and avoid the origin being the global optimum, respectively.

In this paper, we set the number of dimensions to 2000 for all test functions. Since the rotation operation[16] has a quadratic computational complexity, we need to efficiently utilize the shared memory of each node to avoid expensive data-transfer operations, following our previous paper [76]. Note that this setting is critical for the speedup of memory-costly function evaluations for distributed EAs. For all test functions, the initial search range is set to $(-10,10)^n$.

C. Experimental Setups

On all test functions, each algorithm was totally run 5 times. We set two termination conditions for all algorithms to make fair comparisons as much as possible: 1) when the best-so-far fitness is below *1e-10*; 2) when the maximum runtime exceeds *3* hours. As a result, the total CPU runtime *roughly* equals 6600 (= 44 algorithms*10 functions*5 trails*3 hours) hours (i.e., 275 days) if they are run only on a single machine. To reduce the time-consuming benchmarking process, 10 HPC servers were used in parallel, all of which were also together used to build a (slightly heterogenous) clustering computing platform. There is a total of 400 CPU logic cores and about 340GB memory in our private clustering computing platform. Under page limits, please see online Supplementary Materials for detailed configurations of each HPC server machine.

D. Experimental Results and Analyses

In Fig. 4, the convergence curves on all test functions are drawn for 22 optimizers. To avoid crowd in plotting, all other 22 optimizers are shown in online Supplementary Materials, since all of them are much worser than or close (only on one test function) to our distributed algorithm on these functions.

On five test functions (i.e., $schwefel12$, $discus$, $ellipsoid$, $different\_powers$, and $step$), clearly DCC showed the best convergence rates. For the first forth cases, there are multiple (rather than one predominated) promising search directions. DCC's diversity maintaining of covariance matrix can exploit them in parallel at the meta-level. For the last case, properly setting the global step size on-the-fly is important for fast convergence rate since there exist many plateaus. For DCC, its

[14] Here we have excluded the classical evolutionary programming, since its modern versions for continuous optimization is very similar to ES.

[15] https://github.com/Evolutionary-Intelligence/pypop

[16] Generating the rotation matrix via Schmidt orthogonalization has a cubic time complexity.

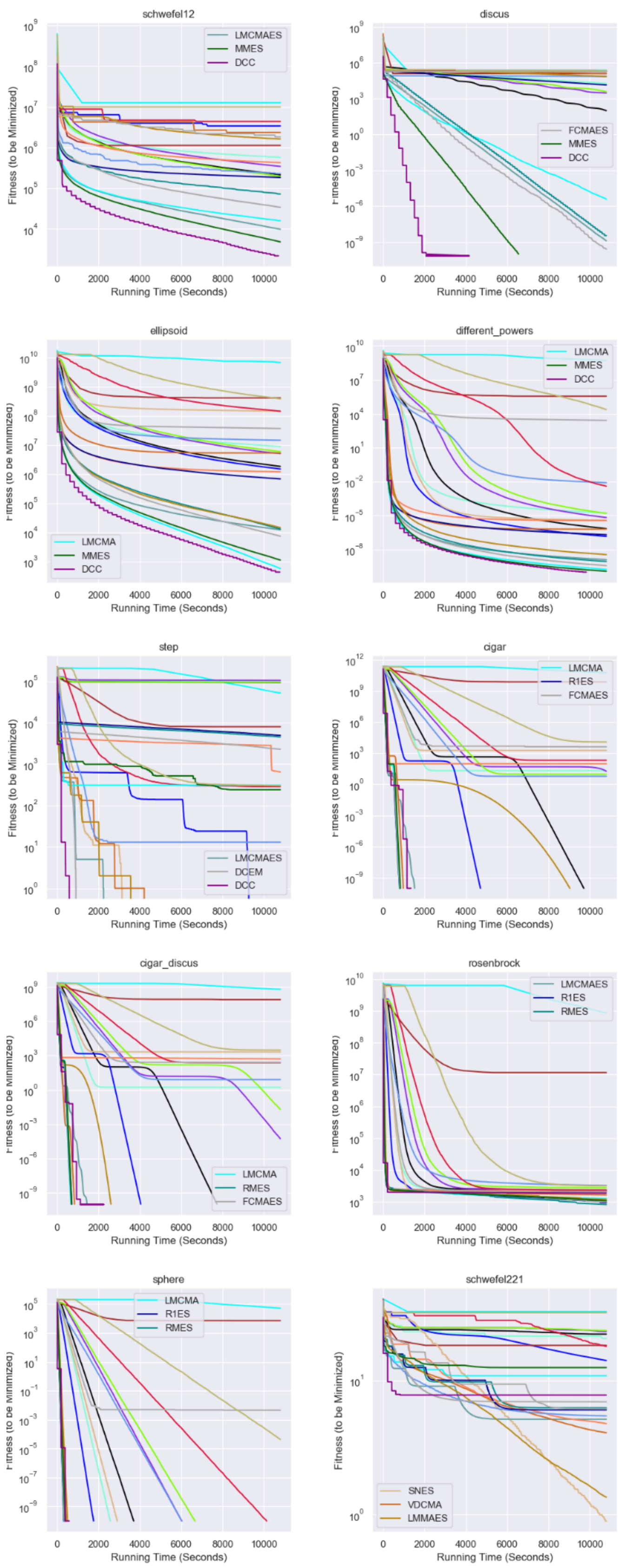


Fig. 4. Median convergence curves on a set of 2000-d rotated and shifted test functions. To avoid crowd and confusion (given 44 algorithms), only the first three of top-ranked optimizers have a legend and only half of optimizers are drawn in each subfigure. Refer to online Supplementary Materials for details.

Meta-ES-based adaptation strategy can alleviate this problem significantly, since it exploits the spatial information well.

On four test functions (i.e., $cigar$, $sphere$, $cigar\_discus$, and $rosenbrock$), DCC still obtained very competitive results, though not the best one. For all of them (except $sphere$), there is one predominated search direction, which can be efficiently learnt by the (exponential smoothing) evolution path. In this case, keeping the diversity on covariance matrix seems to be not important. The performance differences on these functions are small (mainly from the overhead of distributed computing), which may be negligible. For DCC, the stabilization strategy for distribution mean update helps to maintain the advantages of the underlying suboptimizers.

However, distributed computing is **not a panacea**, which cannot escape from the No-Free-Lunch theorems [132]. On $schwefel221$, DCC suffered from a slow convergence. This is due to the *loss of gradient* on this *min-max* function. See online Supplementary Materials for a theoretical analysis.

Overall, the zig-zag-type convergence rate is reminiscent of a famous natural evolution theory "**punctuated equilibrium**" [146]. Such a punctuated equilibrium-style convergence curve may be an essential property of many distributed EAs, which is worthwhile to be further investigated.

VI. Conclusion

In this paper, we have for the first time examined the *coherence* of the theoretical concept (PAS) for CC with many real-world applications, which is believed to be of central importance [113], and showed that *few* real-world applications could well match such an ideal assumption[17]. Instead, we have provided a pure Nash equilibrium (PNE) perspective as a new avenue to capture the essence of CC [147] to explain currently empirical successes on some (rather all) non-separable LSO problems. Furthermore, based on the above theoretical results, we have extended the serial version of CC to the powerful distributed computing scenario and proposed its distributed version (DCC), where collective learning [77], [142] plays an important role when combined with two state-of-the-art ES variants (LM-CMA and CMA-ES). Numerical experiments have shown the scalability (w.r.t. CPU cores) and potentials of DCC on a set of high-dimensional benchmark functions.

We plan to extend DCC in the future as follows: 1) to build a model to analyze its convergence rate; 2) to further validate its efficiency and effectiveness on some real-world applications; 3) to scale it up to more than one thousands of CPU cores ("*more is different*" [148]).

References

[1]LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. *Nature*, 521(7553), pp.436-444.
[2]Schmidhuber, J., 2015. Deep learning in neural networks: An overview. *NN*, 61, pp.85-117.
[3]Zador, A., Richards, B., Ölveczky, B., et al., 2022. Toward next-generation artificial intelligence: Catalyzing the NeuroAI revolution. *arXiv* preprint arXiv:2210.08340.
[4]Nesterov, Y. and Spokoiny, V., 2017. Random gradient-free minimization of convex functions. *FoCM*, 17(2), pp.527-566.

[17] See https://tinyurl.com/3d4nnneb for a variety of applications regarding evolutionary computation.

[5]Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T. and Brockhoff, D., 2021. COCO: A platform for comparing continuous optimizers in a black-box setting. *OMS*, 36(1), pp.114-144.
[6]Salimans, T., Ho, J., Chen, X., Sidor, S. and Sutskever, I., 2017. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv* preprint arXiv:1703.03864.
[7]Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., et al., 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540), pp.529-533.
[8]Fan, J.X., Shen, S.Z., Erwin, D.H., et al., 2020. A high-resolution summary of Cambrian to Early Triassic marine invertebrate biodiversity. *Science*, 367(6475), pp.272-277.
[9]Jaderberg, M., Dalibard, V., Osindero, S., et al., 2017. Population based training of neural networks. *arXiv* preprint arXiv:1711.09846.
[10]Jaderberg, M., Czarnecki, W.M., Dunning, I., et al., 2019. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science*, 364(6443), pp.859-865.
[11]https://www.deepmind.com/blog/how-evolutionary-selection-can-train-more-capable-self-driving-cars (Last Visit: 2023-02-15.)
[12]Forrest, S., 1993. Genetic algorithms: Principles of natural selection applied to computation. *Science*, 261(5123), pp.872-878.
[13]Eiben, A.E. and Smith, J., 2015. From evolutionary computation to the evolution of things. *Nature*, 521(7553), pp.476-482.
[14]Miikkulainen, R. and Forrest, S., 2021. A biological perspective on evolutionary computation. *Nature Machine Intelligence*, 3(1), pp.9-15.
[15]Varelas, K., Auger, A., Brockhoff, D., Hansen, N., ElHara, O.A., Semet, Y., Kassab, R. and Barbaresco, F., 2018, September. A comparative study of large-scale variants of CMA-ES. In *PPSN* (pp. 3-15). Springer.
[16]Omidvar, M.N., Li, X. and Yao, X., 2022. A review of population-based metaheuristics for large-scale black-box global optimization—Part I. *IEEE TEVC*, 26(5), pp.802-822.
[17]Omidvar, M.N., Li, X. and Yao, X., 2022. A review of population-based metaheuristics for large-scale black-box global optimization—Part II. *IEEE TEVC*, 26(5), pp.823-843.
[18]Nesterov, Y., 2018. Lectures on convex optimization. Springer.
[19]Potter, M.A. and De Jong, K.A., 1994. A cooperative coevolutionary approach to function optimization. In *PPSN* (pp. 249-257). Springer.
[20]Potter, M.A., 1997. The design and analysis of a computational model of cooperative coevolution. *Ph.D. dissertation*, George Mason University.
[21]Potter, M.A. and Jong, K.A.D., 2000. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *ECJ*, 8(1), pp.1-29.
[22]Gandomi, A.H., Deb, K., Averill, R.C., Rahnamayan, S. and Omidvar, M.N., 2023. Variable functioning and its application to large scale steel frame design optimization. *SMO*, 66(1), pp.1-17.
[23]Yang, Z., Tang, K. and Yao, X., 2008. Large scale evolutionary optimization using cooperative coevolution. *IS*, 178(15), pp.2985-2999.
[24]Ma, X., Li, X., Zhang, Q., Tang, K., Liang, Z., Xie, W. and Zhu, Z., 2018. A survey on cooperative co-evolutionary algorithms. *IEEE TEVC*, 23(3), pp.421-441.
[25]Hansen, N., Müller, S.D. and Koumoutsakos, P., 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *ECJ*, 11(1), pp.1-18.
[26]Akimoto, Y. and Hansen, N., 2020. Diagonal acceleration for covariance matrix adaptation evolution strategies. *ECJ*, 28(3), pp.405-435.
[27]Ollivier, Y., Arnold, L., Auger, A. and Hansen, N., 2017. Information-geometric optimization algorithms: A unifying picture via invariance principles. *JMLR*, 18(18), pp.1-65.
[28]Hansen, N. and Auger, A., 2014. Principled design of continuous stochastic search: From theory to practice. Theory and Principled Methods for the Design of Metaheuristics, pp.145-180.
[29]Beyer, H.G. and Schwefel, H.P., 2002. Evolution strategies–A comprehensive introduction. *NC*, 1(1), pp.3-52.
[30]Wierstra, D., Schaul, T., Glasmachers, T., Sun, Y., Peters, J. and Schmidhuber, J., 2014. Natural evolution strategies. *JMLR*, 15(1), pp.949-980.
[31]Akimoto, Y., Nagata, Y., Ono, I. and Kobayashi, S., 2012. Theoretical foundation for CMA-ES from information geometry perspective. *Algorithmica*, 64(4), pp.698-716.
[32]Omidvar, M.N., Li, X., Mei, Y. and Yao, X., 2014. Cooperative co-evolution with differential grouping for large scale optimization. *IEEE TEVC*, 18(3), pp.378-393.
[33]Mühlenbein, H. and Mahnig, T., 1999. FDA-A scalable evolutionary algorithm for the optimization of additively decomposed functions. *ECJ*, 7(4), pp.353-376.
[34]Tang, K., Li, X., Suganthan, P.N., Yang, Z. and Weise, T., 2010. Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization. *TR*.
[35]Li, X., Tang, K., Omidvar, M.N., Yang, Z., Qin, K. and China, H., 2013. Benchmark functions for the CEC'2013 special session and competition on large-scale global optimization. *TR*.
[36]Omidvar, M.N., Li, X. and Tang, K., 2015. Designing benchmark problems for large-scale continuous optimization. *IS*, 316, pp.419-436.
[37]Bonabeau, E., Dorigo, M. and Theraulaz, G., 2000. Inspiration for optimization from social insect behaviour. *Nature*, 406(6791), pp.39-42.
[38]Feng, F., Zhang, C., Zhang, S. and Zhang, Q.J., 2016. Parallel decomposition approach to gradient-based EM optimization. *IEEE TMTT*, 64(11), pp.3380-3399.
[39]Cheney, N., Bongard, J., SunSpiral, V. and Lipson, H., 2018. Scalable co-optimization of morphology and control in embodied machines. *JRSI*, 15(143), p.20170937.
[40]Farahmand, A.M., Ahmadabadi, M.N., Lucas, C. and Araabi, B.N., 2010. Interaction of culture-based learning and cooperative co-evolution and its application to automatic behavior-based system design. *IEEE TEVC*, 14(1), pp.23-57.
[41]Vidal, F.P., Lutton, E., Louchet, J. and Rocchisani, J.M., 2010. Threshold selection, mitosis, and dual mutation in cooperative co-evolution: Application to medical 3D tomography. In *PPSN* (pp. 414-423). Springer.
[42]De Rainville, F.M., Sebag, M., Gagné, C., Schoenauer, M. and Laurendeau, D., 2013, July. Sustainable cooperative coevolution with a multi-armed bandit. In *GECCO* (pp. 1517-1524).
[43]Zhai, Y., Ong, Y.S. and Tsang, I.W., 2016. Making trillion correlations feasible in feature grouping and selection. *IEEE TPAMI*, 38(12), pp.2472-2486.
[44]He, S., Jia, G., Zhu, Z., Tennant, D.A., Huang, Q., Tang, K., Liu, J., Musolesi, M., Heath, J.K. and Yao, X., 2016. Cooperative co-evolutionary module identification with application to cancer disease module discovery. *IEEE TEVC*, 20(6), pp.874-891.
[45]Fan, J. and Wang, J., 2016. A collective neurodynamic optimization approach to nonnegative matrix factorization. *IEEE TNNLS*, 28(10), pp.2344-2356.
[46]Gong, M., Liu, J., Qin, A.K., Zhao, K. and Tan, K.C., 2020. Evolving deep neural networks via cooperative coevolution with backpropagation. *IEEE TNNLS*, 32(1), pp.420-434.
[47]Rashid, A.B., Ahmed, M., Sikos, L.F. and Haskell-Dowland, P., 2022. Anomaly detection in cybersecurity datasets via cooperative co-evolution-based feature selection. *ACM TMIS*, 13(3), pp.1-39.
[48]Liu, S., Wang, H., Peng, W. and Yao, W., 2022. A surrogate-assisted evolutionary feature selection algorithm with parallel random grouping for high-dimensional classification. *IEEE TEVC*, 26(5), pp.1087-1101.
[49]Zhao, T.F., Chen, W.N., Kwong, S., Gu, T.L., Yuan, H.Q., Zhang, J. and Zhang, J., 2021. Evolutionary divide-and-conquer algorithm for virus spreading control over networks. *IEEE TCYB*, 51(7), pp.3752-3766.
[50]James, G., Witten, D., Hastie, T. and Tibshirani, R., 2013. An introduction to statistical learning. Springer.
[51]Gomez, F., Schmidhuber, J. and Miikkulainen, R., 2008. Accelerated neural evolution through cooperatively coevolved synapses. *JMLR*, 9(31), pp.937-965.
[52]https://evotorch.ai/ (Last Visit: 2023-2-16.)
[53]Schmidhuber, J., Wierstra, D., Gagliolo, M. and Gomez, F., 2007. Training recurrent networks by evolino. *NCJ*, 19(3), pp.757-779.
[54]Gomez, F.J. and Schmidhuber, J., 2005, June. Co-evolving recurrent neurons learn deep memory POMDPs. In *GECCO* (pp. 491-498). ACM.
[55]Fan, J., Lau, R. and Miikkulainen, R., 2003. Utilizing domain knowledge in neuroevolution. In *ICML* (pp. 170-177).
[56]Gomez, F.J. and Miikkulainen, R., 1999, July. Solving non-Markovian control tasks with neuroevolution. In *IJCAI* (pp. 1356-1361).
[57]Moriarty, D.E. and Mikkulainen, R., 1996. Efficient reinforcement learning through symbiotic evolution. *ML*, 22(1), pp.11-32.
[58]Moriarty, D.E. and Miikkulainen, R., 1995. Efficient learning from delayed rewards through symbiotic evolution. In *ICML* (pp. 396-404). Morgan Kaufmann.
[59]Wright, S.J., 2015. Coordinate descent algorithms. *MP*, 151(1), pp.3-34.
[60]Loshchilov, I., 2017. LM-CMA: An alternative to L-BFGS for large-scale black box optimization. *ECJ*, 25(1), pp.143-171.
[61]Liu, D.C. and Nocedal, J., 1989. On the limited memory BFGS method for large scale optimization. *MP*, 45(1-3), pp.503-528.
[62]Sun, Y., Li, X., Ernst, A. and Omidvar, M.N., 2019, June. Decomposition for large-scale optimization problems with overlapping components. In *IEEE-CEC* (pp. 326-333). IEEE.

[63]Conn, A.R., Gould, N.I.M. and Toint, P.L., 1989. An introduction to the structure of large-scale nonlinear optimization problems and the LANCELOT project. *TR*.

[64]Lipson, H. and Pollack, J.B., 2000. Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799), pp.974-978.

[65]Panait, L., Tuyls, K. and Luke, S., 2008. Theoretical advantages of lenient learners: An evolutionary game theoretic perspective. *JMLR*, 9, pp.423-457.

[66]Chen, B.H., Wang, T.Z., Li, C.T., Dai, H.J. and Song, L., 2021. Molecule optimization by explainable evolution. In *ICLR*.

[67]Greff, K., Van Steenkiste, S. and Schmidhuber, J., 2017. Neural expectation maximization. In *NeurIPS*, 30.

[68]Leiserson, C.E., Thompson, N.C., Emer, J.S., et al., 2020. There's plenty of room at the Top: What will drive computer performance after Moore's law?. Science, 368(6495), p.eaam9744.

[69]Nesterov, Y., 2012. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIOPT*, 22(2), pp.341-362.

[70]Richtárik, P. and Takáč, M., 2016. Distributed coordinate descent method for learning with big data. *JMLR*, 17(1), pp.2657-2681.

[71]Shi, H.J.M., Tu, S., Xu, Y. and Yin, W., 2016. A primer on coordinate descent algorithms. arXiv preprint arXiv:1610.00040.

[72]Ratliff, L.J., Burden, S.A. and Sastry, S.S., 2016. On the characterization of local Nash equilibria in continuous games. *IEEE TAC*, 61(8), pp.2301-2307.

[73]Panait, L., 2010. Theoretical convergence guarantees for cooperative coevolutionary algorithms. *ECJ*, 18(4), pp.581-615.

[74]Wiegand, R.P., 2003. An analysis of cooperative coevolutionary algorithms. *Ph.D. dissertation*, George Mason University.

[75]Wiegand, R.P. and Sarma, J., 2004, September. Spatial embedding and loss of gradient in cooperative coevolutionary algorithms. In *PPSN* (pp. 912-921). Springer.

[76]Duan, Q., Zhou, G., Shao, C., Yang, Y. and Shi, Y., 2022, August. Collective learning of low-memory matrix adaptation for large-scale black-box optimization. In *PPSN* (pp. 281-294). Springer.

[77]Vanchurin, V., Wolf, Y.I., Katsnelson, M.I. and Koonin, E.V., 2022. Toward a theory of evolution as multilevel learning. *PNAS*, 119(6), p.e2120037119.

[78]Hansen, N., Ros, R., Mauny, N., Schoenauer, M. and Auger, A., 2011. Impacts of in-variance in search: When CMA-ES and PSO face ill-conditioned and non-separable problems. *ASC*, 11(8), pp.5755-5769.

[79]Van den Bergh, F. and Engelbrecht, A.P., 2004. A cooperative approach to particle swarm optimization. *IEEE TEVC*, 8(3), pp.225-239.

[80]Chen, W., Weise, T., Yang, Z. and Tang, K., 2010. Large-scale global optimization using cooperative coevolution with variable interaction learning. In *PPSN* (pp. 300-309). Springer.

[81]Li, X. and Yao, X., 2012. Cooperatively coevolving particle swarms for large scale optimization. *IEEE TEVC*, 16(2), pp.210-224.

[82]Mei, Y., Omidvar, M.N., Li, X. and Yao, X., 2016. A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. *ACM TOMS*, 42(2), pp.1-24.

[83]Sabar, N.R., Abawajy, J. and Yearwood, J., 2016. Heterogeneous cooperative co-evolution memetic differential evolution algorithm for big data optimization problems. *IEEE TEVC*, 21(2), pp.315-327.

[84]Yang, M., Omidvar, M.N., Li, C., Li, X., Cai, Z., Kazimipour, B. and Yao, X., 2016. Efficient resource allocation in cooperative co-evolution for large-scale global optimization. *IEEE TEVC*, 21(4), pp.493-505.

[85]Omidvar, M.N., Yang, M., Mei, Y., Li, X. and Yao, X., 2017. DG2: A faster and more accurate differential grouping for large-scale black-box optimization. *IEEE TEVC*, 21(6), pp.929-942.

[86]Ge, H., Sun, L., Tan, G., Chen, Z. and Chen, C.P., 2017. Cooperative hierarchical PSO with two stage variable interaction reconstruction for large scale optimization. *IEEE TCYB*, 47(9), pp.2809-2823.

[87]Sun, Y., Kirley, M. and Halgamuge, S.K., 2017. A recursive decomposition method for large scale continuous optimization. *IEEE TEVC*, 22(5), pp.647-661.

[88]Ren, Z., Liang, Y., Zhang, A., Yang, Y., Feng, Z. and Wang, L., 2018. Boosting cooperative coevolution for large scale optimization with a fine-grained computation resource allocation strategy. *IEEE TCYB*, 49(12), pp.4180-4193.

[89]Wang, Y., Liu, H., Wei, F., Zong, T. and Li, X., 2018. Cooperative coevolution with formula-based variable grouping for large-scale global optimization. *ECJ*, 26(4), pp.569-596.

[90]Peng, X., Jin, Y. and Wang, H., 2018. Multimodal optimization enhanced cooperative coevolution for large-scale optimization. *IEEE TCYB*, 49(9), pp.3507-3520.

[91]Yang, M., Zhou, A., Li, C. and Yao, X., 2020. An efficient recursive differential grouping for large-scale continuous problems. *IEEE TEVC*, 25(1), pp.159-171.

[92]Liu, H., Wang, Y. and Fan, N., 2020. A hybrid deep grouping algorithm for large scale global optimization. *IEEE TEVC*, 24(6), pp.1112-1124.

[93]Xu, P., Luo, W., Lin, X., Zhang, J., Qiao, Y. and Wang, X., 2021. Constraint-objective cooperative coevolution for large-scale constrained optimization. *ACM TELO*, 1(3), pp.1-26.

[94]Chen, A., Ren, Z., Guo, W., Liang, Y. and Feng, Z., 2022. An efficient adaptive differential grouping algorithm for large-scale black-box optimization. *IEEE TEVC*. Early Access.

[95]Ma, X., Huang, Z., Li, X., Wang, L., Qi, Y. and Zhu, Z., 2022. Merged differential grouping for large-scale global optimization. *IEEE TEVC*, 26(6), pp.1439-1451.

[96]Wu, Y., Peng, X., Wang, H., Jin, Y. and Xu, D., 2022. Cooperative coevolutionary CMA-ES with landscape-aware grouping in noisy environments. *IEEE TEVC*. Early Access.

[97]Kumar, A., Das, S. and Mallipeddi, R., 2022. An efficient differential grouping algorithm for large-scale global optimization. *IEEE TEVC*. Early Access.

[98]Xu, P., Luo, W., Lin, X., Chang, Y. and Tang, K., 2022. Difficulty and contribution based cooperative coevolution for large-scale optimization. *IEEE TEVC*. Early Access.

[99]Mitchell, M., Holland, J. and Forrest, S., 1993. When will a genetic algorithm outperform hill climbing. In *NeurIPS* (pp. 51-58).

[100]Chen, W.N., Jia, Y.H., Zhao, F., Luo, X.N., Jia, X.D. and Zhang, J., 2019. A cooperative co-evolutionary approach to large-scale multisource water distribution network optimization. *IEEE TEVC*, 23(5), pp.842-857.

[101]Chen, M., Du, W., Tang, Y., Jin, Y. and Yen, G.G., 2022. A decomposition method for both additively and non-additively separable problems. *IEEE TEVC*. Early Access.

[102]Li, J.Y., Zhan, Z.H., Tan, K.C. and Zhang, J., 2022. Dual differential grouping: A more general decomposition method for large-scale optimization. *IEEE TCYB*. Early Access.

[103]Komarnicki, M.M., Przewozniczek, M.W., Kwasnicka, H. and Walkowiak, K., 2022. Incremental recursive ranking grouping for large scale global optimization. *IEEE TEVC*. Early Access.

[104]Zhang, X.Y., Gong, Y.J., Lin, Y., Zhang, J., Kwong, S. and Zhang, J., 2019. Dynamic cooperative coevolution for large scale optimization. *IEEE TEVC*, 23(6), pp.935-948.

[105]Jia, Y.H., Mei, Y. and Zhang, M., 2020. Contribution-based cooperative co-evolution for nonseparable large-scale problems with overlapping subcomponents. *IEEE TCYB*, 52(6), pp.4246-4259.

[106]Zhang, X., Ding, B.W., Xu, X.X., Li, J.Y., Zhan, Z.H., Qian, P., Fang, W., Lai, K.K. and Zhang, J., 2022. Graph-based deep decomposition for overlapping large-scale optimization problems. *IEEE TSMC-S*. Early Access.

[107]Wiegand, R.P., Liles, W.C. and De Jong, K.A., 2002. Modeling variation in cooperative coevolution using evolutionary game theory. In *FOGA* (pp. 203-220).

[108]Ficici, S.G., Melnik, O. and Pollack, J.B., 2005. A game-theoretic and dynamical-systems analysis of selection methods in coevolution. *IEEE TEVC*, 9(6), pp.580-602.

[109]Jansen, T. and Wiegand, R.P., 2003, July. Exploring the explorative advantage of the cooperative coevolutionary (1+ 1) EA. In *GECCO* (pp. 310-321). Springer.

[110]Jansen, T. and Wiegand, R.P., 2004. The cooperative coevolutionary (1+ 1) EA. *ECJ*, 12(4), pp.405-434.

[111]Toint, P.L., 1983. Test problems for partially separable optimization and results for the routine PSPMIN. *TR*.

[112]Porcelli, M. and Toint, P.L., 2022. Exploiting problem structure in derivative free optimization. *ACM TOMS*, 48(1), pp.1-25.

[113]Averick, B.M., Carter, R.G., Xue, G. L. and Moré, J.J., 1992. The MINPACK-2 test problem collection. *TR*, Argonne National Laboratory.

[114]Bouaricha, A. and Morè, J. J., 1997. Impact of partial separability on large-scale optimization. *COA*, 7, 27-40.

[115]Hildreth, C., 1954. Point estimates of ordinates of concave functions. *JASA*, 49(267), pp.598-619.

[116]https://oeis.org/A000110/list (Last Visit: 2023-2-17.)

[117]Nash, J.F., 1950. Equilibrium points in n-person games. *PNAS*, 36(1), pp.48-49.

[118]Nash, J.F., 1951. Non-cooperative games. *AM*, pp.286-295.

[119]Duan, Q., Shao, C., Qu, L., et al., 2019, June. When cooperative co-evolution meets coordinate descent: Theoretically deeper understandings and practically better implementations. In *IEEE-CEC* (pp. 721-730). IEEE.

[120]Shang, Y.W. and Qiu, Y.H., 2006. A note on the extended Rosenbrock function. *ECJ*, 14(1), pp.119-126.

[121]Warga, J., 1963. Minimizing certain convex functions. *JSIAM*, 11(3), pp.588-593.

[122]Rosen, J.B., 1965. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica*, pp.520-534. 76.

[123]Tatarenko, T. and Kamgarpour, M., 2018. Learning generalized Nash equilibria in a class of convex games. *IEEE TAC*, 64(4), pp.1426-1439.

[124]Lanctot, M., Zambaldi, V., Gruslys, A., et al., 2017. A unified game-theoretic approach to multiagent reinforcement learning. In *NeurIPS* (pp. 4190-4203).

[125]Hespanha, J.P., 2017. Noncooperative game theory: An introduction for engineers and computer scientists. Princeton University Press.

[126]Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets. In *NeurIPS* (pp. 2672-2680).

[127]Schmidhuber, J., 2020. Generative adversarial networks are special cases of artificial curiosity (1990) and also closely related to predictability minimization (1991). *NN*. 127, pp.58-66.

[128]Hildreth, C., 1957. A quadratic programming procedure. *NRLQ*, 4(1), pp.79-85. (Erratum: https://doi.org/10.1002/nav.3800040410)

[129]D'Esopo, D.A., 1959. A convex programming procedure. *NRLQ*, 6(1), pp.33-42.

[130]Powell, M.J., 1973. On search directions for minimization algorithms. *MP*, 4(1), pp.193-201.

[131]Whitley, D., Rana, S., Dzubera, J. and Mathias, K.E., 1996. Evaluating evolutionary algorithms. *AIJ*, 85(1-2), pp.245-276.

[132]Wolpert, D.H. and Macready, W.G., 1997. No free lunch theorems for optimization. *IEEE TEVC*, 1(1), pp.67-82.

[133]Leon, S.J., Linear algebra with applications (8th ed). Upper Saddle River, NJ: Pearson/Prentice Hall, 2010.

[134]Boyd, S.P. and Vandenberghe, L., 2004. Convex optimization. Cambridge University Press.

[135]Zaharia, M., Xin, R.S., Wendell, P., et al., 2016. Apache spark: A unified engine for big data processing. *CACM*, 59(11), pp.56-65.

[136]Moritz, P., Nishihara, R., Wang, S., et al., 2018. Ray: A distributed framework for emerging AI applications. In *OSDI* (pp. 561-577).

[137]Holland, J.H., 1962. Outline for a logical theory of adaptive systems. *JACM*, 9(3), pp.297-314.

[138]Fogel, D.B., 1999. An overview of evolutionary programming. In Evolutionary Algorithms (pp. 89-109). Springer, New York, NY.

[139]Kennedy, J. and Eberhart, R., 1995, November. Particle swarm optimization. In *ICNN* (pp. 1942-1948). IEEE.

[140]Storn, R. and Price, K., 1997. Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces. *JGO*, 11(4), pp.341-359.

[141]Larrañaga, P. and Lozano, J.A. eds., 2001. Estimation of distribution algorithms: A new tool for evolutionary computation. Springer Science & Business Media.

[142]Schwefel, H.P., 1988. Collective intelligence in evolving systems. In Ecodynamics (pp. 95-100). Springer, Berlin, Heidelberg.

[143]Loshchilov, I., Glasmachers, T. and Beyer, H.G., 2019. Large scale black-box optimization by limited-memory matrix adaptation. *IEEE TEVC*, 23(2), pp.353-358.

[144]Rudolph, G., 1990, October. Global optimization by means of distributed evolution strategies. In *PPSN* (pp. 209-213). Springer.

[145]Kudela, J., 2022. A critical problem in benchmarking and analysis of evolutionary computation methods. *NMI*, 4(12), pp.1238-1245.

[146]Gould, S.J. and Eldredge, N., 1993. Punctuated equilibrium comes of age. *Nature*, 366(6452), pp.223-227.

[147]De Jong, E.D., Stanley, K.O. and Wiegand, R.P., 2007, July. Introductory tutorial on coevolution. In *GECCO* (pp. 3133-3157). ACM.

[148]Anderson, P.W., 1972. More is different: Broken symmetry and the nature of the hierarchical structure of science. *Science*, 177(4047), pp.393-396.