A review of population-based metaheuristics for large-scale black-box global optimization: Part A

Mohammad Nabi Omidvar, *Senior Member, IEEE,* Xiaodong Li, *Fellow, IEEE,* and Xin Yao, *Fellow, IEEE*

***Abstract*—Scalability of optimization algorithms is a major challenge in coping with the ever growing size of optimization problems in a wide range of application areas from high-dimensional machine learning to complex large-scale engineering problems. The field of large-scale global optimization is concerned with improving the scalability of global optimization algorithms, particularly population-based metaheuristics. Such metaheuristics have been successfully applied to continuous, discrete, or combinatorial problems ranging from several thousand dimensions to billions of decision variables. In this two-part survey, we review recent studies in the field of large-scale black-box global optimization to help researchers and practitioners gain a bird's-eye view of the field, learn about its major trends, and the state-of-the-art algorithms. Part A of the series covers two major algorithmic approaches to large-scale global optimization: problem decomposition and memetic algorithms. Part B of the series covers a range of other algorithmic approaches to large-scale global optimization, describes a wide range of problem areas, and finally touches upon the pitfalls and challenges of current research and identifies several potential areas for future research.**

***Index Terms*—large-scale global optimization, black-box optimization, metaheuristics, evolutionary optimization**

I. Introduction

The curse of dimensionality is "a malediction that has plagued the scientists from the earliest days" [1] and taming it has been at the heart of many research efforts in computational sciences ranging from computational linear algebra [2] and machine learning [3] to numerical optimization [4]. The prime motive in all these areas is to devise new ways of building scalable computational systems capable of "doing more with less".

In the context of numerical optimization, the curse of dimensionality is caused by the exponential growth in the size of the search space with respect to an increase in the number of input variables. In recent years, this has been loosely referred to as "large-scale optimization" or "large-scale global optimization". The term *global* is to emphasize the role of heuristics and metaheuristics, especially in the context of continuous optimization. It should be noted that the notion of "large-scale" changes over time and varies from problem to problem. In this paper, a problem is considered large-scale if it causes scalability issues on the state-of-the-art algorithms.

More specifically, a single objective optimization problem can be defined as follows (assuming minimization):

$$\min \ f(\mathbf{x}),\ \mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{A} \tag{1}$$

$$s.t.:\ \mathbf{g}(\mathbf{x}) \leq 0 \tag{2}$$

$$\mathbf{h}(\mathbf{x}) = 0, \tag{3}$$

where $\mathcal{A}$ is the domain of the function f and $\mathbf{x}$ is an n-dimensional vector in $\mathcal{A}$, and $\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), \cdots, g_p(\mathbf{x}))$ and $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \cdots, h_q(\mathbf{x}))$ are vectors of inequality and equality constraints respectively. Large-scale optimization is concerned with the scalability of optimization algorithms as n grows in size and its effect on the number of constraints and their dimensionality.

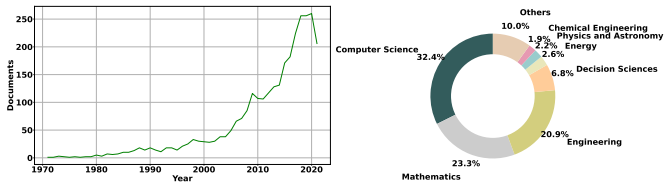
Rapid technological advancements causes the emergence of ever growing optimization problems in various areas. For example, in construction engineering we are entering the so-called "era of the megatall buildings" with the construction of the first kilometer-tall building already underway [5]. This has resulted in large-scale optimization problems [6] in construction engineering. Similarly, the data explosion phenomenon has caused the emergence of large-scale optimization problems at the heart of many data analytics and learning problems [3, 7]. Advances in machine learning and the rise of deep artificial neural networks has also resulted in optimization problems with over a billion variables [8, 9]. These optimization problems not only grow in size but do so in an exponential manner, i.e., the number of decision variables they entail also grows exponentially [10]. This rapid growth has stimulated scientific research in various areas to build scalable optimization algorithms. Figure 1a clearly shows the rapid growth in the number of scientific articles published on large-scale optimization in the last decade. Figure 1b also shows the contribution of different subject areas to the topic. The dominance of Computer Science and Mathematics is an indication of the importance of the algorithmic aspects of designing efficient optimization methods.

Population-based metaheuristics have also gained popularity in recent years for solving large-scale global optimization problems [11]. Despite the common criticism of being computationally expensive, the ubiquity of parallel computing has rendered the issue of population size and generational cost of secondary nature in light of their unique capacity in dealing with multimodal landscapes, deceptive functions, and their general search capability. It has recently been

Mohammad Nabi Omidvar is with the School of Computing, University of Leeds, and Leeds University Business School, Leeds LS2 9JT, UK (email: m.n.omidvar@leeds.ac.uk). He is also the current chair of the IEEE Taskforce on Large-Scale Global Optimization.

Xiaodong Li is with the School of Computing Technologies, RMIT University, Melbourne, Australia (email: xiaodong.li@rmit.edu.au).

Xin Yao is with the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China. Xin Yao is also with the School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK (email: x.yao@cs.bham.ac.uk; xiny@sustc.edu.cn).



(a) (b)

Fig. 1: Publication trend in large-scale global optimization from 1971 to 2021. The results show the number of documents containing variations of the phrases "large scale optimization" or "large scale global optimization" in their title, abstract, or keywords. The results also include the hyphenated version and also apply to multi- and many-objective algorithms. Source: Scopus.

shown that evolutionary algorithms, a type of population-based metaheuristics, can rival classic optimization methods that have dominated the field of deep learning [12]. Evolutionary algorithms have also shown great capacity in solving problems of millions or even billions of variables where their classic counterparts proved inefficient [13–17].

In dealing with very complex problems, we are deemed to resort to two main strategies: (*a*) to find an exact solution to a simplified model of a given problem; (*b*) to find an approximate solution to a complex but more accurate model of a given problem. Therefore, using a less competent optimization (or search) algorithm either demands over simplification of a given problem, or results in a low-quality solution to a more accurate model of the problem. In the context of large-scale global optimization, developing better search algorithms has two implications: (*a*) we can start to tackle even larger (more complex) problems; (*b*) we can find better solutions to the problem of the same size. In recent years, a wide range of methods have been considered for large-scale optimization. These often fall within one of the following themes:

- Problem decomposition.
- Hybridization, memetic algorithms, and local search.
- Sampling and variation operators.
- Approximation and surrogate modeling.
- Initialization.
- Parallelization.

In this two-part survey, we give a comprehensive review of large-scale global optimization looking into each of the above themes, summarizing the main research findings, and discussing their advantages and disadvantages. The scope and the breadth of topics we cover in this series makes it distinct from other review works in the field [11, 18]. Areas such as large-scale constrained optimization and large-scale multiobjective optimization have become active in the last two years, which are almost absent from other reviews. They also categorize the large-scale algorithms into decomposition and non-decomposition methods. Based on the main approaches stated above, in this paper we give a more nuanced taxonomy of approaches to large-scale global optimization.

Another unique feature of this paper is that it looks at

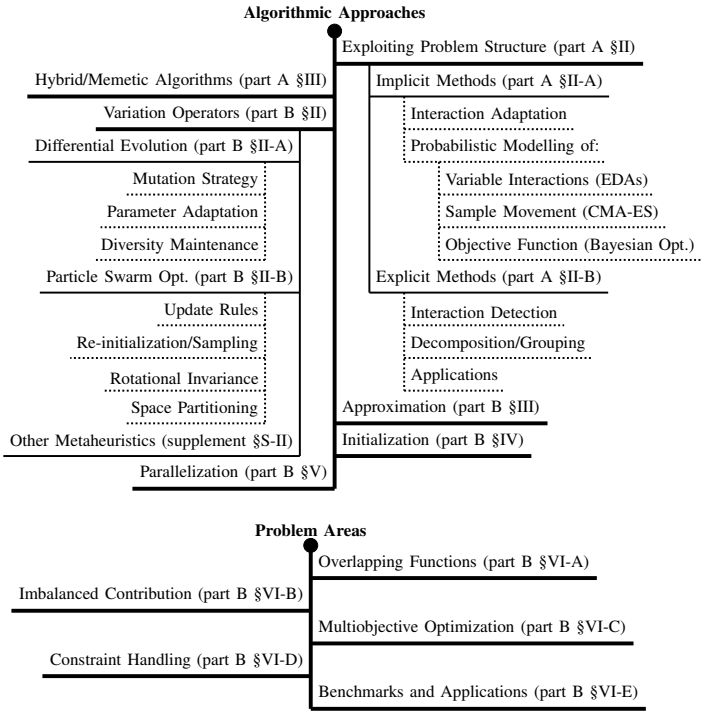
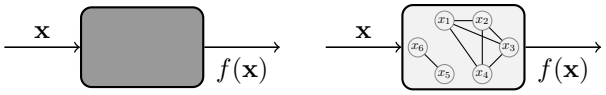


Fig. 2: Outline of the topics covered in the two parts of this survey series on large-scale global optimization.

the above themes from a variable interaction perspective. We believe that the efficiency of algorithms is largely dependent on their effectiveness in exploiting problem structure. Variable interaction is an important attribute of optimization problems with a direct effect on a problem's degree of nonlinearity, its overall structure, and degree of modularity. In classic genetic algorithms (GA) research for example, tight linkage is central to their scalability [19]. The design of an inversion operator is also motivated by the importance of tight linkage to minimize the disturbance of interacting genes by the crossover operator. In memetic algorithms, variable interaction affects the efficiency of dimension-wise local search [20, 21]. In cooperative coevolution and other divide-and-conquer methods, variable interaction governs the utility of a decomposition [22]. In estimation of distribution algorithms and other related algorithms, their rotational invariance is determined by how well the interaction information are captured within their covariance matrix [23]. These are just a few examples to emphasize the extensive impact of variable interaction. Part B of this survey also features a section on pitfalls and challenges of large-scale optimization, open research questions, and other special topics such as large-scale multi-objective optimization, and large-scale dynamic optimization.

Outline: This survey series comes in two parts and a supplementary document accompanying part A. The two parts jointly cover the six approaches listed above as well as five major problem areas including overlapping functions, imbalanced contribution, multiobjective optimization, constraint handling, and benchmarks and applications. Figure 2 depicts a high-level structure of the main topics covered across both parts. Part A covers problem decomposition (§II) and hybridization,



(a) (b)

Fig. 3: A black-box optimization problem can be converted to a grey-box problem by means of variable interaction analysis or incorporation of other sources of information such as analyzing its constraints or domain knowledge.

memetic algorithms and local search (§III), which are two of the most widely researched approaches in the field. Part B covers the remaining algorithmic approaches, several major problem areas, and a section on the pitfalls and challenges of large-scale global optimization, and some suggestions for future research. The background material is covered in the supplementary document accompanying part A.

II. Linkage Learning and Exploiting Problem Structure

The guiding principle of the algorithms presented in this section is to exploit the hidden or clouded structure of a given problem. Exploiting problem structure is common in many branches of optimization [24–27]. Grey-box optimization is a relatively new concept referring to the study of incorporating problem structure into the optimization process [28] (see Figure 3). This notion of finding and exploring the "hidden order" [29] is an old one in evolutionary computation and has been studied extensively in the context of linkage learning [30, 31]. Goldberg et al. [32] showed that linkage plays a crucial role in the performance of GAs. Even separable problems can be exponentially difficult for simple GAs if the variable dependence information is unknown [32, 33]. It has also been shown that GAs with no linkage learning mechanism requires an exponentially growing population size in order to locate the global optimum Thierens and Goldberg [34]. In the continuous domain, the rotation of the fitness landscape, which has the effect of introducing linkage between decision variables, changes the time complexity of GAs from $\mathcal{O}(n \log n)$ to $\mathcal{O}(n^n)$ [35]. It is therefore clear that in high-dimensional spaces, for an algorithm to be computationally plausible, incorporation of structural information is paramount.

In grey-box optimization, it is assumed that the problem structure is known *a priori*. This is particularly the case for a wide range of discrete and combinatorial problems. For example, it is not realistic to assume that nothing is known about a travelling salesman problem (TSP) problem therefore treating it as strictly black-box [36, 37]. The same goes for other popular discrete problems such as gene sequencing [38], or pseudo-boolean problems such as MAX-kSAT or CNF-SAT [39]. Constraints of a problem can also reveal some information about its structure. For example, variable reduction strategy [40, 41] allows the explicit use of constraint functions to infer some relationships among the variables and formulate some variables as functions of a set of core variables. In many problems however, particularly in the continuous domain, the structure may not be evident. Therefore, to exploit problem structure, it first needs to be discovered. Many algorithms have been proposed for discovering problem structure in the form of capturing its variable interaction topology. The merits of these algorithms are not limited to converting black-box problems into grey boxes. There are studies showing that even when the problem formulation is fully known (white box) and the dimensionality is not necessarily high, variable interaction analysis methods can help reveal nontrivial information about the problem [42].

There are also several different forms in which the structural information can be used to enhance the optimization performance. Some methods such as cooperative coevolution [43] decompose the problem into a number of *explicit* lower-dimensional subproblems, therefore, requiring a variable interaction analysis method to form a plausible problem decomposition. Some other methods such as estimation of distribution algorithms (EDAs) [44, 45] and Bayesian optimization algorithms [46] do not decompose the problem per se. They instead *implicitly* capture and make use of the interaction information in a probabilistic model of the problem they construct during the optimization process. In the rest of this section, we review such methods in the context of large-scale optimization. Figure 4 gives an overview of implicit and explicit methods of exploiting problem structure. The section that follows covers the implicit methods and Section II-B covers the explicit methods.

A. Implicit Methods

Implicit methods exploit problem structure by building and evolving a model of the problem, which can then be used to bias the search towards promising regions of the search space. These methods differ in their choice of model representation and the information from which the model is created.

Interaction Adaptation: These methods are extensions of simple genetic algorithms with added mechanisms to promote tight linkage in problem representation. The simplest of such mechanisms is the inversion operator [47], which acts on a string of variables and changes their order to increase the likelihood of placing the interacting variables next or close to one another. Although the reordering of variables (genes) have shown to improve the performance of GAs [48], they are extremely slow in generating tight linkages [49]. This clearly limits their applicability to large-scale optimization.

Other methods such as the messy GA (mGA) [32], fast messy GA (fmGA) [50], gene expression messy GA (gemGA) [51], linkage learning genetic algorithm (LLGA) [52], and linkage evolving genetic operator (LEGO) [53] use more sophisticated representations than a simple bit string to encode linkage information into the representation and employ special operators to change the representation over time to promote tightly linked representations.

Probabilistic Models: These methods use a probability distribution to represent the objective function or its various characteristics. Estimation of distribution algorithms (EDAs) [44, 54] compact genetic algorithm (cGA) [55], Bayesian optimization algorithms (BOAs) [46, 56], and

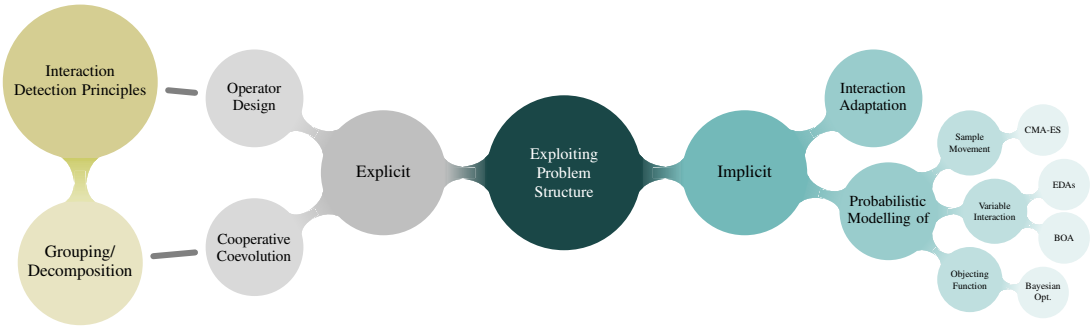


Fig. 4: Exploiting problem structure can be done implicitly (§II-A) by adapting the problem representation or probabilistic modeling, or explicitly (§II-B) by means of variable interaction analysis (§II-B1) and problem decomposition (§II-B2).

Bayesian optimization [57] are examples of such methods. The high-level working principle of these methods is to generate one or more sample solutions from a probability distribution. The generated solution(s) are then used to update the parameters of the probabilistic model in an iterative process. Many different versions of such probabilistic optimization algorithms exist in the literature, which differ in the choice of their probabilistic models and how they are updated. Below is a short summary of three major types.

- Building a model of variable interactions: estimation of distribution algorithms (EDAs) [44, 54], and Bayesian optimization algorithms (BOAs) [46] are two important such algorithms that capture variable interactions in their probabilistic models. BOAs are different from Bayesian Optimization which builds a model of the objective function and is discussed in the next bullet point. EDAs with a multivariate normal distribution can represent the interaction by adapting the covariance matrix of the Gaussian distribution. Different variants of EDAs have been used for large-scale global optimization, which will be covered later in this section. BOA uses Bayesian networks to represent interactions. To do so, the algorithm needs to learn the structure of the Bayesian network and its parameters (conditional probabilities). This method is very flexible and efficient in representing and solving complex interaction patterns such as overlapping [58] or hierarchical [58] problems. However, the model selection process which involves learning the structure of the Bayesian network is an NP-hard problem, which makes BOA a poor choice for large-scale global optimization.
- Building a model of the objective function: In Bayesian optimization [57], not to be confused with BOA, the unknown objective function is treated as a random function modeled by placing a prior distribution over it. Whenever the actual objective function is evaluated, its input/output pair is used as new evidence to update the prior distribution to form the posterior distribution of the objective function. Finally, an acquisition function is constructed from the posterior distribution, which is used to determine the location of the next query point. Scalability of Bayesian optimization is the subject of several recent studies [59–67]; however, a detailed review of such techniques is out of the scope of this paper. For a review of Bayesian optimization the reader is referred to the paper by Shahriari et al. [57].
- Building a model of the population movement: covariance matrix adaptation evolution strategy (CMA-ES) [68] is a popular model building algorithm which uses a multivariate Gaussian distribution to model the so-called *successful steps* taken by the algorithm during the course of optimization. In differential evolution (DE), its contour matching property [69], which is similar to the notion of modeling population movement, the step sizes and their orientations are adapted to the landscape of the objective function. Details of advances in DE literature on large-scale optimization is given in part B of the survey.

Estimation of Distribution Algorithms: A major problem of EDAs is their scalability issue in solving large-scale problems. Three major reasons contribute to this scalability issue: (a) Accurate estimation of the distribution of high quality solutions requires a relatively large sample size which grows exponentially with the dimensionality of the problem [70]; (b) A small sample size will result in poor estimation of the eigenvalues of the covariance matrix [71]; and (c) the cost of sampling from a multi-dimensional Gaussian distribution increases cubically with problem size [72].

The strategies to alleviate the scalability issue of EDAs can be categorized into two major types: 1) Space partitioning and dimensionality reduction methods where the aim is to control the complexity of the covariance matrix; and 2) Use of heavy-tail distributions to improve exploration and diversity.

Space partitioning and dimensionality reduction: EDAs with univariate models [73, 74] treat an n-dimensional problem as n 1-dimensional problems and as such are the simplest and have the most efficient sampling. However, several theoretical [75, 76] and empirical [77] studies have shown that univariate EDAs are inadequate in solving non-separable problems. A full multivariate model on the other hand can be computationally expensive in high-dimensional spaces.

Dong et al. [78] propose an EDA with model complexity control which applies a threshold to the global Pearson correlation matrix to find weakly correlated dimensions and models them with univariate distributions and partitions the remaining strongly correlated variables into a set of lower dimensional spaces each of which is modeled using a multivariate distribution. To alleviate the deficiencies of Pearson correlation

with non-Gaussian samples, Xu et al. [79] proposed to use mutual information instead to detect variable dependence. A major downside of using mutual information however is its high computational cost [22]. Space partitioning by means of random grouping and cooperative coevolution has been used in several other studies [80–83] to manage the complexity of EDAs.

Dimensionality reduction techniques and space projection are other alternatives to control the model complexity of multivariate EDAs. Kabán et al. [84] borrowed the idea of random projection to lower dimensions from the theory of random matrices to tackle the scalability issue of EDAs [84, 85]. They proposed an algorithm that randomly projects the original large-scale problem into an ensemble of lower-dimensional problems. The random matrix theory suggests that with an appropriate choice of target dimensions, it is possible to preserve important features of the original space (such as Euclidean distances or dot products) in the reduced space within a reasonable tolerance [86]. It has also been shown that the distribution of the sample points become more Gaussian in the reduced space [87]. These features makes it possible to capture the variable correlation of the high-dimensional space using a lower-dimensional subspaces; therefore, making the parameter estimation of EDAs more viable using less computational resources. This eliminates the need for over simplification of the model in EDAs as is the case in univariate EDAs. It is clear that random projection to a lower dimensional space is not unique, and sample points can be projected down into any subspace of the original space. For this reason, Kabán et al. [84] use an ensemble of projected points and estimate the covariance of the sample points in each lower-dimensional subspace. Finally, the ensemble of projections are used to construct a new population (sample) in the original space. It has been shown that a proper combination of the ensemble of projected points results in a natural smoothing effect that ensures the exploration capability of the algorithm. In a similar vein, Dong et al. [88] used PCA to transform the multivariate Gaussian model of EDA into its principal lower-dimensional latent subspace.

Heavy-tail Distributions: Sampling based on heavy-tail distributions have been used in many population-based algorithms [89–91] with the aim of improving exploration and population diversity. A range of such distributions including Lévy, Cauchy, and t-distributions have been used to enhance the performance of EDAs [73, 92, 93]. Among these, the Cauchy distribution has been used more widely with EDAs. Although the literature is clear on the efficacy of Cauchy sampling on low-dimensional problem [94, 95], there is controversy in its utility on high-dimensional problems [96]. Hansen et al. [97] reported that Cauchy's long jumps are almost invariably ineffective, while other studies found it beneficial [73, 74]. Sanyang et al. [92] compared the performance of univariate and multivariate Cauchy distributions with the Gaussian distribution within a random projection framework on a range of large-scale problems with up to 1000 dimensions. They reported that although a multivariate Cauchy performs better than a univariate Cauchy, they both perform worse than a multivariate Gaussian distribution on large-scale problems. In another subsequent study, they provided a theoretical explanation for the poor performance of Cauchy distribution on large-scale problems, which was shown to be consistent with empirical results [96]. The study showed that unlike Gaussian norms, Cauchy norms lack a good concentration property causing a disproportionate number of very large steps, which results in an inefficient search strategy as the dimensions increase.

Scalability of CMA-ES: Covariance Matrix Adaptation Evolution Strategy [98] is a popular optimization algorithm which approximates the contours of a given objective function by means of a Gaussian distribution. CMA-ES exhibits several invariance properties including rotation invariance, which is central for efficient optimization of nonseparable functions. This is achieved through iterative updating of a covariance matrix used to control its underlying sampling Gaussian distribution. CMA-ES has two major limitations in dealing with high-dimensional problems [99]: 1) High number of strategy parameters which is the degrees of freedom in the covariance matrix and scales quadratically with the dimension. In other words, the space complexity of CMA-ES is $\mathcal{O}(n^2)$; and 2) High computational complexity that comes from the operations needed to adapt the covariance matrix, i.e., sampling from a multivariate normal distribution, updating of the covariance matrix and its factorization and eigen-decomposition. This results in a cubic complexity in the number of dimensions $\mathcal{O}(n^3)$.

There have been several attempts to improve the scalability of CMA-ES by reducing its time and space complexities. Igel et al. [100] replaced the eigen-decomposition update rule of CMA-ES with the Choleskey decomposition resulting in reducing its time complexity to $\mathcal{O}(n^2)$. Poland and Zell [101] proposed the adaptation of the most significant mutation vector in Main Vector Adaptation Evolution Strategy (MVA-ES), which reduces the time complexity of the algorithm to linear. Knight and Lunacek [102] restrained the optimization of an n-dimensional problem to its m main components. This algorithm, L-CMA-ES, reduces the time complexity to $\mathcal{O}(m^2n)$ and the space complexity to $\mathcal{O}(mn)$. For the case of $m = 1$ the algorithm reduces to MVA-ES, and for $m = n$ it reduces to CMA-ES. Sun et al. [103] proposed another linear time evolution strategy, R1-NES, which uses a low-rank approximation of the covariance matrix by only considering its predominant eigen-direction. Ros and Hansen [99] proposed sep-CMA-ES in which only the diagonal elements of the covariance matrix is adopted. This reduces the time complexity of the algorithm to $\mathcal{O}(n)$. LM-CMA [104, 105] is another limited memory implementation of CMA-ES inspired by the classic L-BFGS [106]. LM-CMA combines the idea of a low-rank approximation of the covariance matrix with Choleskey decomposition to reduce the time and space complexity of the algorithm to $\mathcal{O}(mn)$. Inspired by LM-CMA, Li et al. [107] proposed the fast CMA-ES for large-scale optimization, which maintains a number of evolution paths and uses two to generate new solutions.

Beyer and Sendhoff [108] proposed matrix adaptation ES to avoid the construction of the covariance matrix by replacing it with an overall transformation matrix involving

only matrix-matrix and matrix-vectors operations. This has reduced the time complexity of the algorithm to $\Theta(n^3/\log(n))$. Loshchilov et al. [109] further improved the time and space complexity to $\Theta(n^2)$ by replacing the multiplicative updates with additive ones.

Li and Zhang [110] proposed to reduce the time and space complexity of CMA-ES by restricting the covariance matrix to a specific simple form. More specifically, they suggested the following form: $\mathbf{C} = \alpha\mathbf{S} + \mathbf{L}$, where $\mathbf{C}$ is the covariance matrix, $\mathbf{S}$ is a sparse matrix, $\mathbf{L}$ is a symmetric low-rank matrix, and $\alpha > 0$. When $\mathbf{S} = \mathbf{I}$, and $\mathbf{L}$ is a rank-one matrix, the algorithm becomes a rank-one ES (R1-ES). By controlling the rank of $\mathbf{L}$, the model can be generalized to rank-m ES (Rm-ES). He et al. [111] proposed to completely replace the Gaussian sampling by a Gaussian mixture model which exhibits richer variable interaction modeling capability.

B. Explicit Methods

Explicit methods capture problem structure information into explicit forms such as variable interaction matrices or trees and use them to either decompose the problem into a set of lower dimensional subproblems [43], or design special variation operators, such as crossover, that respect the problem structure [112, 113]. Unlike implicit methods, explicit methods require extra objective function evaluations to find the variable interaction structure.

One popular explicit method, which has gained popularity in large-scale global optimization, is the cooperative coevolution (CC) [43]. The CC framework requires the problem to be decomposed into a set of lower dimensional subproblems each of which is optimized separately. The CC framework maintains a separate population for each subproblem (a.k.a component) which are "coevolved" in a round-robin fashion. Since the candidate solutions to each component do not form a complete solution, representative solutions of other components are required to form a complete solution for evaluation. These representative solutions form a complete solution known as the *context vector* [114] which is used to evaluate all partial solutions. The context vector is updated iteratively and acts as the context in which the *cooperation* occurs.

The first design choice in using CC is problem decomposition. It is clear that this can be performed in many different ways. The first CC algorithm [43], cooperative coevolution genetic algorithm (CCGA), decomposes an n-dimensional problem into n 1-dimensional problems, where n is the problem dimension. CCGA was used to solve problems with up to 30 dimensions. Liu et al. [115] made the first attempt to solve large-scale optimization problems using a CC framework. They used fast evolutionary programming as the component optimizer in a CC framework with the decomposition strategy of CCGA to solve problems with up to 1000 dimensions [115].

van den Bergh and Engelbrecht [114] suggested that full decomposition strategy of CCGA runs the risk of introducing pseudominima, i.e., "minima created as a side effect of the partitioning of the search space". This is consistent with the observation that CCGA does not perform well on problems with interacting decision variables such as Griewank and Rosenbrock test functions [43]. To alleviate this problem, van den Bergh and Engelbrecht [114] proposed the use of PSO with a k s-dimensional decomposition instead of the extreme n 1-dimensional decomposition used by CCGA. Another decomposition strategy, divide-in-half, was proposed by Shi et al. [116] where the problem is divided into two equally-sized components which are optimized iteratively using differential evolution [117].

It is clear that the algorithms discussed so far are oblivious to variable interaction and may place interacting variables in different components. This has a detrimental effect on the optimization performance and makes the algorithm sensitive to the updating policy of the context vector. A function can be decomposed in many different ways without any knowledge of the underlying variable interaction structure. It is therefore important to form the components such that the interaction between them are kept to a minimum. In the next section, we review several algorithms that attempt to deal with the variable interaction problem.

1) Dealing with Variable Interaction: This section is devoted to the review of various techniques to address variable interaction. Most of the methods presented here are concerned with an accurate identification of variable interactions, which are subsequently used to decompose a problem into its constituent parts. Decomposition of a problem often has two components: (i) A mechanism to identify interactions between the decision variables (covered in this section). (ii) A mechanism to form a set of subproblems based on the interaction information (§II-B2). It is often the case that the interaction detection mechanism necessitates a certain way of forming the groups. For example, some methods provide the interaction information for every pair of variables in the form of an interaction matrix. This matrix contains sufficient information about the number of components and their sizes to warrant an automatic decomposition. Some other methods rely on various heuristics that increase the likelihood of placing interacting variables close to one another. Such methods do not suggest an obvious decomposition of the problem, therefore requiring extra information such as the number of components and their sizes to be supplied by the user.

In this paper, we identified seven interaction detection principles and three decomposition mechanisms. Figure 5 shows this classification and the interplay between them. Table S-I of the supplementary document contains a list of specific algorithms belonging to each class and its corresponding decomposition strategy. In what follows, each detection principle and decomposition mechanism is studied in further details.

a) Random Grouping: Random grouping [118] is the most basic way of dealing with variable interaction in cooperative coevolution. The rationale behind it is to randomly arrange the decision variables after each coevolutionary cycle to increase the probability of placing interacting variables into the same component. Despite being superior to an arbitrary static decomposition [119], random grouping has two major drawbacks. Firstly, the user has to decide about the number and the size of each component. Secondly, the probability of simultaneously placing several interacting variables in one components approaches zero when there are many such vari-

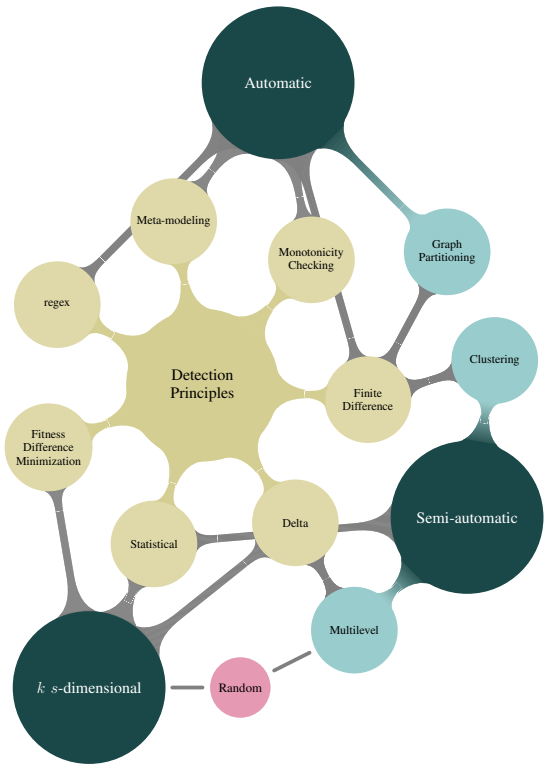
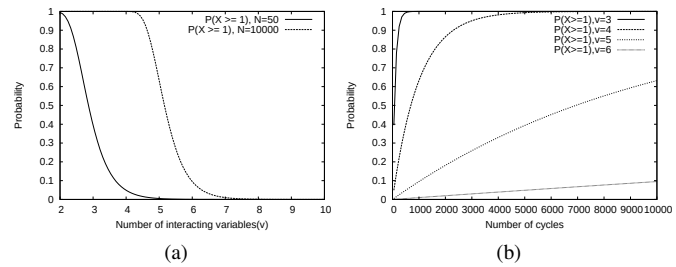
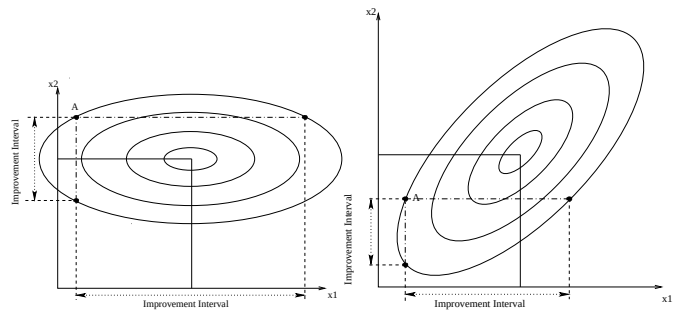


Fig. 5: Common variable interaction detection principles (olive) and variable grouping strategies (dark teal) used by explicit decomposition algorithms. The links indicate which grouping principles are common among which detection principles. The light teal color indicates variation on the grouping strategies.



(a) (b)

Fig. 6: Random grouping's likelihood of correctly grouping interacting variables ($m = 10$).



(a) (b)

Fig. 7: Shrinkage of improvement intervals under coordinate rotation of the same landscape.

ables. More specifically, given N cycles, the probability of assigning v interacting variables into one of the m available components for at least k cycles is [120]:

$$P(X \geq k) = \sum_{r=k}^{N} \binom{N}{r} \left(\frac{1}{m^{v-1}}\right)^{r} \left(1 - \frac{1}{m^{v-1}}\right)^{N-r} \quad (4)$$

Equation (4) shows that the probability of placing v variables in one component for at least k cycles decreases geometrically as v increases. Fig. 6a shows the sharp decline in probability correctly grouping interacting variables as the number of such variables increase. The figure also shows that a 200-fold increase in the number of random reordering can hardly accommodate two extra variables with the same probability. Fig. 6b shows how the probability of correctly grouping $v = 3$ to $v = 6$ interacting variables increases with the number of trials; however, this does not significantly increase the likelihood of a correct grouping when the number of variables is high.

b) Delta Grouping: An alternative grouping approach called *delta grouping* [121] was shown to outperform random grouping on most functions from a set of 20 large-scale benchmark problems [122]. The rationale behind delta grouping is that the improvement interval of nonseparable variables are relatively smaller than those of separable variables [123] (Fig. 7). Therefore, delta grouping sorts the variables based on the average dimension-wise displacement of the sample points between two consecutive iterations. Once the decision variables are sorted based on the magnitude of their average displacement called 'delta' (δ), they are grouped into k components of size s both of which are determined by the user. A major drawback of delta grouping is its low performance on functions with more than one nonseparable component. Ge et al. [124] improved the grouping quality of delta grouping by measuring the success and failure rate of groups and evolving the grouping accordingly.

Liu et al. [125] used the idea of delta grouping to solve nonseparable functions. The algorithm that they proposed uses line search along each dimension to estimate the improvement along each dimension. The dimensions with similar improvements are then grouped together to form a subproblem for optimization. Unlike delta grouping which is used with cooperative coevolution, this method uses local search and a modified DE which applies the mutation operator to a designated subset of decision variables belonging to the same group.

c) Fitness Difference Minimization: The methods reviewed in this section are based on adaptive rearrangement of decision variables to minimize an error function, which is claimed to minimize the interaction between resultant components [126]. The rationale is that for a partially separable function, the difference between the overall objective function and the sum of its nonseparable subfunctions should be zero, i.e., $f(\mathbf{x}) - \sum_{i=1}^{m} f_i(\mathbf{x}_i) \equiv 0$, where $f_i(\mathbf{x}_i)$ is the ith nonseparable subfunction. Motivated by this, Sayed et al. [126]

decompose a problem by finding an arrangement of decision variables that minimizes an approximation to the following least square equation:

$$\min \left[f(\mathbf{x}) - \sum_{i=1}^{m} f_i(\mathbf{x}_i)\right]^2. \tag{5}$$

Due to the black-box assumption, the number of component sizes and their dimensions are unknown. Therefore, Sayed et al. [126] assume a uniform k d-dimensional components and search of a rearrangement of the decision variable that minimizes the following equation:

$$\min \left[m\left(f(\mathbf{c}_1) + f(\mathbf{c}_2)\right) - \sum_{i=1}^{k} \left\{\hat{f}_i(\mathbf{c}_1;\mathbf{c}_2) + \hat{f}_i(\mathbf{c}_2;\mathbf{c}_1)\right\}\right]^2. \tag{6}$$

This equation is an approximation to (5) where $\mathbf{c}_1$ and $\mathbf{c_2}$ are solutions whose elements are set to constants c_1 and c_2 respectively, and $\hat{f}_i(\mathbf{x};\mathbf{y})$ is a parameterized version of f with the variables belonging to the ith component set to $\mathbf{x}$ and the rest to $\mathbf{y}$. This variable grouping method, which is called dependency identification (DI) was shown to outperform random grouping on the CEC'2010 large-scale benchmark suite. Aguilar-Justo and Mezura-Montes [127] replaced the random rearrangement with two more systematic strategies to generate variations in the grouping which showed better performance as compared to the random case. A problem of all decomposition methods based on fitness difference minimization is that they require the user to specify the number or the size of each component. To fix this problem in the context of constrained problems, Aguilar-Justo et al. [128] proposed to evolve the best arrangement of the decision variables as well as the number of components using GA.

Sayed et al. [129] also proposed a variation of (6) where sum of absolute differences is used instead of sum of squares. Despite DI's improved performance as compared to random grouping, the optimization problem defined in (6) is NP-hard due to large number of possible k d-dimensional decompositions. To alleviate this problem, Sayed et al. [126] randomly rearrange only 10% of the decision variables with greedy search. Dai et al. [130] conducted a study on the effect this parameter on the performance of DI. The experimental results showed that DI is sensitive to this parameter with a tendency toward a better performance when the rate is larger than 60%. The results also confirmed the superiority of DI over random grouping on a wide range of rates; however, this was not the case when compared to delta grouping.

d) Statistical Methods: The methods reviewed in this section rely on statistical features of the evolving population to infer variable interaction. Tiwari et al. [131] proposed the idea of using regression analysis to deal with what they call *inseparable function interaction* and *variable dependence*. Inseparable function interaction is identical to what we call variable interaction or epistasis in this paper. Variable dependence however pertains to the existence of functional dependence between decision variables, i.e., the possibility of expressing one variable as a function of some other set of variables. Although the two notions appear to be linked, the authors did not investigate their connection from a formal mathematical point of view. Following the idea of variable dependence, Tiwari and Roy [132] proposed the genetic algorithm for variable dependence (GAVD) which uses regression analysis to find a set of dependent and independent decision variables by iteratively setting regression coefficients of individual decision variables to zero and examining the goodness of fit. Once the independent and dependent decision variables are found, GA is used to optimize the independent set and regression analysis is used to estimate proper values for the dependent set. Roy and Tiwari [133] proposed the generalized regression GA (GRGA) to deal with inseparable function interaction. They employed regression analysis on the decision variables in the course of optimization and study the changes in the coefficient of the regression model over time to guide the optimization process. A major drawback of both GAVD and GRGA is poor scalability.

One statistical way of dealing with variable interactions is to calculate the Pearson correlation matrix of the population and use a threshold on the coefficients to form the components. Here the assumption is that weak correlations indicate weak interactions. These techniques often calculate the correlations based on either the entire population or its top p% samples. Correlation-based adaptive variable partitioning (AVP) Ray and Yao [134] is one such method which groups pairs of variables whose correlations are larger than a predefined threshold and optimizes them against the remaining set in a co-evolutionary manner. Singh and Ray [135] proposed an improved version, AVP2, which can form multiple groups of various sizes depending on the underlying variable interaction structure. For each variable $i \in \{1, \ldots, n\}$, AVP2 forms a group with all other decision variables whose correlation coefficient with the ith variable is above a certain threshold. This results in a total of n potentially overlapping groups which are subsequently merged based on their common variables to form a set of disjoint groups. Rojas and Landa [136] proposed another correlation-based decomposition, 4CDE, which calculates the correlation coefficient of each variable and the objective function and divides the resulting correlation coefficients into equally-sized intervals. The variables whose correlation coefficient fall within the same interval form a component which is subsequently optimized in a CC framework with differential evolution as its component optimizer. This process is repeated and the correlation coefficients are updated using exponential smoothing.

Some algorithms use the dimension-wise variance magnitudes of the population to form the groups. In variance priority cooperative coevolution, Wang et al. [137] form the groups based on the variance magnitude of the current candidate solutions along various dimensions and selects the top k variables having the largest variances to form a component. Liu and Tang [83] proposed a cooperative coevolutionary framework based on CMA-ES, which adaptively selects from a pool of three decomposition strategies at each coevolutionary cycle. The three decomposition methods are: random grouping, Min-Variance Decomposition (MiVD), and Max-Variance Decomposition. The rationale behind using various decompositions is to regulate the exploration/exportation balance of the algorithm. MiVD sorts the variables based on the magnitude

of the main diagonal elements of CMA-ES's covariance matrix and divides them into k s-dimensional groups. This strategy minimizes the intra-group variances. Conversely, MaVD maximizes the intra-group variance and forms the groups by taking a variable from every group formed by MiVD which is also used in variance priority CC proposed by Wang et al. [137]. To coordinate between various decomposition methods, the algorithm assigns a probability value to each decomposition method at the end of each cycle, which is used to randomly select a decomposition method for the next cycle.

All the statistical methods presented so far cannot detect interacting variables with a reasonable precision and rely on a user to specify the number and/or the size of components. For example, although the use of variance magnitudes has its own merits and can potentially improve the optimization performance, its effectiveness in capturing variable interaction has not been established. The Pearson correlation can only capture linear relationships among the variables, which is also inaccurate for measuring variable interaction. To address these issues, Sun et al. [22] proposed the maximum entropic epistasis (MEE) which uses maximal information coefficient [138] to check for functional relationship between a variable i and the partial derivative of the objective function with respect to another variable j, i.e. $\frac{\partial f}{\partial x_j}$. Since MIC is based on mutual information, it can capture nonlinear relationships. To deal with the black-box nature of the problem, MEE approximates $\frac{\partial f}{\partial x_j}$ using finite differences: $\frac{\partial f}{\partial x_j} \approx \frac{f(x_j+\delta x_j)-f(x_j)}{\delta x_j}$. MEE uses this method to check for *direct* interactions between all pairs of variables by applying a threshold on the MIC values to form a binary variable interaction matrix. Finally, the interaction matrix is treated as the adjacency matrix of an undirected graph and the groups are formed by finding independent components using the breadth-first search algorithm. MEE has shown a very high variable interaction detection accuracy on an array of 24 high-dimensional functions. Two major drawbacks of MEE is its high computational cost caused by pair-wise analysis of the decision variables, and its sensitivity to the choice of threshold value.

e) Meta-modelling: The methods discussed in this section infer variable interaction information in the process of building a surrogate or a meta-model for the objective function. Although meta-modelling is often used for expensive function optimization, there has been little or no attempts to use it for finding problem structure. One such algorithm is proposed by Mahdavi et al. [139], which uses high-dimensional model representation (HDMR) [140] technique to find interacting variables. HDMR has the following general form that can be used to approximate a function:

$$f(\mathbf{X}) = f_0 + \sum_{i=1}^{n} f_i(x_i) + \sum_{1\le i\le j\le n} f_{ij}(x_{ij}) + f_{1\ldots n}(x_1,\ldots,x_n), \tag{7}$$

where f_0 is the zeroth order term, $f_i(x_i)$ is the first-order terms capturing the effect of each variable acting independently, f_{ij} is the second-order term capturing the correlated contribution of x_i and x_j, and finally, $f_{1\ldots n}$ is the nth-order term capturing the joint correlation of all decision variables not covered by all other terms. HDMR has a finite number of terms and is exact once all terms are included. A nice property of HDMR is that if the contribution of a pth order terms are negligible, the contribution of all higher order term will be smaller. Mahdavi et al. [139] used a particular type of HDRM called RBF-HDMR [141] to approximate the objective function up to the second-order terms in order to capture variable interaction between pairs of variables. This new technique is capable of finding the number of components and their sizes with good accuracy and has shown good performance on the CEC'2010 large-scale benchmark suite. In another study, Li et al. [142] used cut-HDRM [143] to find separable and nonseparable variables which are used in turn to approximate a given high-dimensional function using support vector regression HDMR.

f) Monotonicity Detection: Munetomo and Goldberg [144] were first to propose a variable interaction detection method by checking the violation of monotonicity conditions through systematic perturbation of the objective function. The monotonicity conditions are defined as follows [144]:

$$\text{if } f(\mathbf{s}^{(i)}) > f(\mathbf{s}) \text{ and } f(\mathbf{s}^{(j)}) > f(\mathbf{s}) \text{ then } f(\mathbf{s}^{(ij)}) > f(\mathbf{s}^{(i)}) \text{ and } f(\mathbf{s}^{(ij)}) > f(\mathbf{s}^{(j)}) \tag{8}$$

$$\text{if } f(\mathbf{s}^{(i)}) < f(\mathbf{s}) \text{ and } f(\mathbf{s}^{(j)}) < f(\mathbf{s}) \text{ then } f(\mathbf{s}^{(ij)}) < f(\mathbf{s}^{(i)}) \text{ and } f(\mathbf{s}^{(ij)}) < f(\mathbf{s}^{(j)}), \tag{9}$$

where $\mathbf{s}^{(\cdot)}$ denotes a candidate solution vector perturbed at the index specified in the bracket. It is clear that (8) checks for monotonic increase, and (9) checks for monotonic decrease. Munetomo and Goldberg [144] developed an algorithm based on (8) and (9) called linkage identification by non-monotonicity detection (LIMD) which checks for violation of these conditions on a randomly initialized population. Any violation of the above two conditions for variables i and j will declare them as interacting. LIMD was tested on low dimensional binary problems.

In the context of large-scale optimization, Chen et al. [145] proposed cooperative coevolution with variable interaction learning (CCVIL) in which they used a similar principle as was used in LIMD for variable interaction in large-scale continuous problems. They declared two dimensions i and j interact if the following condition holds:

$$\exists\, \mathbf{s}, \mathbf{s}^{(i)}, \mathbf{s}^{(j)}, \mathbf{s}^{(ij)} : f(\mathbf{s}) > f(\mathbf{s}^{(i)}) \wedge f(\mathbf{s}^{(j)}) < f(\mathbf{s}^{(ij)}). \tag{10}$$

Similar to the notation used before, $\mathbf{s}^{(\cdot)}$ denotes the solution $\mathbf{s}$ perturbed at dimensions specified in the parenthesis. Although (10) appears to be different from the monotonicity checks defined by (8) and (9), the algorithmic implementation of CCVIL ensures that the following condition is also satisfied: $f(\mathbf{s}) > f(\mathbf{s}^{(j)})$. CCVIL uses monotonicity checking within a coevolutionary framework proposed by Weicker and Weicker [146] to find disjoint interaction groups of a given objective function. CCVIL starts by assuming full separability among all decision variables. It then iteratively processes all dimensions and checks their interaction with other dimensions. If an interaction is detected the respective dimensions merge to form an interaction group. It should be noted that CCVIL uses a generalization of (10) in which multiple dimensions can be perturbed simultaneously. This allows the algorithm to

check for interaction between a decision variable and an entire interaction group. CCVIL has shown good performance on the CEC'2010 LSGO benchmark suite [122].

It should be noted that satisfaction of the monotonicity conditions for a given set of sample points does not guarantee separability of the decision variables. Indeed, two variables are separable only if the conditions hold for *all* possible choices of the four sample points needed in (8) and (9). Therefore, repeated application of the monotonicity conditions can increase the likelihood that two variables are actually detected as separable. Based on this idea, Sun et al. [147] proposed statistical variable interdependence learning (SVIL) in which the monotonicity conditions are checked for all pairs of variables over k random sets of points. SVIL treats the proportion of detected interactions between the ith and the jth variables as their interaction probability which is stored in a matrix $\mathbf{D}$. To decompose a problem, SVIL applies a threshold on $\mathbf{D}$ to force it into a binary matrix. Since the grouping accuracy of SVIL is sensitive to the choice of the threshold parameter, the algorithm adapts this parameter in the course of optimization. Next, for each decision variable, a group is formed with all other decision variables which interact with it. This means that SVIL forms n overlapping groups where n is the dimensionality of the problem. Finally, SVIL is used in a cooperative coevolution framework to optimize each component in a round-robin fashion. Although the empirical results on high-dimensional CEC'2005 benchmark suite [148] suggest that SVIL is effective in improving the optimization performance, its variable interaction accuracy has not been studied independently using modular benchmarks suite as the CEC'2010 [122] and CEC'2013 [149] large-scale suites.

A major drawback of SVIL is its high computational cost due to its pair-wise interaction detection mechanism between the decision variables, which makes it a quadratic algorithm, i.e., $\mathcal{O}(n^2)$. To alleviate this issue, Ge et al. [150] proposed a generalized version of the monotonicity check in which is simultaneous perturbations of multiple dimensions are allowed to check for interaction between two sets of decision variables $\mathcal{B}_1$ and $\mathcal{B}_2$. This is equivalent of replacing dimension i with $\mathcal{B}_1$ and dimension j with $\mathcal{B}_2$ in (8) and (9). This generalization allows us to check the interaction of a single variable with all other variables with a single application of monotonicity conditions, i.e., $\mathcal{B}_1 = \{x_i\}$ and $\mathcal{B}_2 = \{1, \dots, n\} \setminus \{x_i\}$. Based on this generalization the authors propose a recursive algorithm where the interaction of a single variable (x_i) is checked against a set which is initialized to all other variables. If x_i is found to be separable, the procedure returns; otherwise, the set is recursively divided into smaller sets until its cardinality reaches one. At this stage all function calls return and merge the interacting variables into a group. This reduces the time complexity of the algorithm down to $\mathcal{O}(n \log n)$. For separable functions, the decomposition can happen in linear time in the number of dimensions n.

g) Finite Differences: The methods discussed in this section use finite differences to detect interacting variables. Although not explicitly defined as such, linkage identification by nonlinearity check (LINC) [151] is a variable interaction learning algorithm based on finite differences to find the linkage sets of binary optimization problems. The same mechanism has also been used to identify interacting groups for real-valued problems (LINC-R) [152]. Both LINC and LINC-R were used with multi-population GAs for solving low dimensional problems [151, 153].

Omidvar et al. [154] proposed differential grouping (DG) by deriving a set of finite difference equations for interaction detection from the definition of partially additive functions (see Def. S.3 of the supplementary document) and applied it to large scale problems. Omidvar et al. [154] showed LINC-R equations can be derived from the differential grouping theorem. Despite their algebraic equivalence, DG is less susceptible to computational roundoff errors due to its simpler computations.

Theorem 1 (Omidvar et al. [154])**.** *Let $f(\mathbf{x})$ be an additively separable function. $\forall a, b_1 \neq b_2, \delta \in \mathbb{R}^1, \delta \neq 0$, variables x_p and x_q interact if the following condition holds*

$$\Delta_{\delta,x_p}[f](\mathbf{x})|_{x_p=a,x_q=b_1} \neq \Delta_{\delta,x_p}[f](\mathbf{x})|_{x_p=a,x_q=b_2}, \quad (11)$$

where

$$\Delta_{\delta,x_p}[f](\mathbf{x}) = f(\dots, x_p + \delta, \dots) - f(\dots, x_p, \dots), \quad (12)$$

refers to the forward difference of f with respect to variable x_p with interval δ.

Theorem 1 states that two variables x_p and x_q interact if the result of (12) for a given value of x_p yields different results for different choices of x_q (i.e., b_1 and b_2). Theorem 1 is derived by showing that under the assumption of additive separability, the finite difference functions on the two sides of (11) give the same results, i.e., separability $\implies \Delta^{(1)} = \Delta^{(2)}$. The contraposition of this proposition can be used to detect interactions, i.e., $\Delta^{(1)} \neq \Delta^{(2)} \implies$ nonseparability. Since exact equality cannot be checked on computational systems, the left hand side of the implication is changed to $\lambda = |\Delta^{(1)} - \Delta^{(2)}| > \epsilon$. It should be noted that although $\Delta^{(1)} \neq \Delta^{(2)}$ implies an interaction, their equality does not necessarily imply separability. Indeed the theorem is silent for this case. However, by invoking weak syllogism, one can argue that observing $\Delta^{(1)} = \Delta^{(2)}$ makes separability more plausible. If repeated application of Theorem 1 with different random values results in $\Delta^{(1)} = \Delta^{(2)}$, the probability of the two variables being separable increases exponentially [152]. However, for most practical applications, a single equality observation is sufficiently accurate for finding separable variables. Nonetheless, failing to detect an interaction is more detrimental to the optimization performance than the conservative case of assuming separable variables to be interacting.

Although Theorem 1 can be used to individual interactions between pairs of variables, the theorem itself does not dictate a particular grouping mechanism. Canonical DG works by choosing a candidate variable x_i and scanning all other dimensions to find all interactions with x_i. If an interaction is found the variable is removed from the set being scanned and is grouped with x_i to form a component. The process continues

[1]Values of a, b_1, b_2 and δ are chosen such that f is evaluated within its domain.

until all variables interacting with x_i are extracted and added to the component containing x_i. This procedure assumes full nonseparability within a component, i.e., all variables interact with all other variables. This approach fails on problems with overlapping components. For example, if the following interaction pattern exist: $x_i \leftrightarrow x_j \leftrightarrow x_k$, in the first pass of the algorithm x_j will be removed from the set and the interaction between x_j and x_k will never be checked. Rosenbrock is such a problem on which DG exhibits a poor accuracy [154].

Another grouping strategy is to form an $n \times n$ interaction matrix and form the nonseparable groups (or components) based on analyzing the interaction matrix. Several algorithms are based on this idea and treat the interaction matrix as the adjacency matrix of an undirected graph and use the connected components algorithm to form the groups. Global DG (GDG) [155], graph-based DG (gDG) [156], and DG2 [157] are such methods. These algorithms can potentially give an accurate picture of the problem structure at the expense of having a quadratic time complexity. Among these, DG2 is the most efficient and achieves the theoretical lower bound for the required number of function evaluations to form a complete interaction matrix based on the repeated application of Theorem 1. This lower bound suggests that to improve the efficiency we either need to compromise the accuracy or change the underlying theorem. These two possibilities are addressed next.

Indirect interactions: Sun et al. [158] proposed the notion of indirect interactions (Def. 1) and used it in an algorithm called extended DG (XDG) to improve the grouping efficiency.

Definition 1. *Let $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ be a differentiable function. Decision variables x_i and x_j conditionally (indirectly) interact if for any candidate solution $\mathbf{x}^*$, $\frac{\partial^2 f(\mathbf{x}^*)}{\partial x_i \partial x_j} = 0$, and a set of decision variables $\{x_{k_1}, \ldots, x_{k_t}\} \subset X$ exists, such that $x_i \leftrightarrow x_{k_1} \leftrightarrow \ldots \leftrightarrow x_{k_t} \leftrightarrow x_j$.*

XDG works by forming n components, one for each variable, containing all the variables it interacts with. This means that a variable may appear in multiple components. To reduce the number of function evaluations, XDG relies on Def. 1 and does not check x_i and x_j for interaction if both interact with a common variable x_k. Finally, XDG merges the groups whose intersection is not null. Although XDG saves some function evaluations due to the indirect interaction assumption, it is not as efficient as DG2 and its time complexity remains $\mathcal{O}(n^2)$. Fast interdependency identification (FII) [159] is another algorithm that draws on the notation of indirect interaction to improve the grouping efficiency.

The generalized theorem: Hu et al. [159] were first to use simultaneous perturbations multiple dimensions to check the interaction of any two sets of decision variables using only four function evaluations. This makes it possible to find all separable variables in $\mathcal{O}(n)$ by iteratively checking each variables against all other variables. For nonseparable variables, the algorithm starts with a candidate variable and checks it against all other variables. If an interaction is detected the variables are merged to form a group. Thereafter, the variables formed into the group are perturbed simultaneously to find if any of its members interacts with the next decision variable. Since a group is always checked against a variable, a considerable number of function evaluations can be saved.

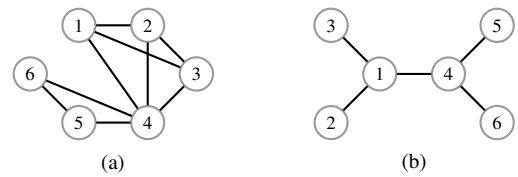


Fig. 8: The interaction structures represented by (a) and (b) cannot be distinguished by RDG, FII, and XDG [157].

Sun et al. [160] formalized the notion of simultaneous perturbations and proposed the following extended version of DG theorem:

Theorem 2 (Sun et al. [160])**.** *Let $f\colon \mathbb{R}^n \to \bar{\mathbb{R}}$ be an objective function and $D = \{1, \ldots, n\}$; $X_1 \subset D$ and $X_2 \subset D$ be two mutually exclusive subsets of decision variables: $X_1 \cap X_2 = \emptyset$. If there exist two unit vectors $\mathbf{u}_1 \in U_{X_1}$ and $\mathbf{u}_2 \in U_{X_2}$, two real numbers $l_1, l_2 > 0$, and a candidate solution $\mathbf{x}^*$ in the decision space, such that*

$$f(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_2) - f(\mathbf{x}^* + l_2\mathbf{u}_2) \neq f(\mathbf{x}^* + l_1\mathbf{u}_1) - f(\mathbf{x}^*), \quad (13)$$

there is at least one interaction between a variable in X_1 and another in X_2.

Theorem 2 implies that with only four function evaluations, the interaction between arbitrary sets X_1 and X_2 can be established. Sun et al. [160] used Theorem 2 to propose recursive differential grouping (RDG) to form the interaction groups. This reduces the time complexity of interaction detection to $\mathcal{O}(n \log n)$ which is lower than the theoretical lower bound based on DG2 [157]. As stated earlier, this reduction in time complexity comes at the expense of losing on the accuracy of a full interaction matrix. As a result, algorithms such as RDG, FII, and XDG cannot detect overlapping components of a function. For instance, these algorithms cannot distinguish the interaction graphs depicted in Fig. 8. Yang et al. [161] further reduced the computational cost of RDG by maintaining and using historical information during the decomposition process to avoid some unnecessary evaluations. Kim and Choi [162] also improved upon RDG by pruning its recursive search three in the divide-and-conquer process. They also used a variable influence metric to pre-sort the decision variables with the aim of increasing the chance of pruning and hence reducing the total number of objective function evaluations. In another study, Xue et al. [163] reduced the depth of RDG's recursion tree by dividing the variables into three subsets [164] instead of two, thus reducing the number of objective function evaluations. Xue et al. [163] proposed an alternative view of the simultaneous perturbations and augmented it with the topological information of the problem to further reduce the computational cost of problem decomposition.

Interaction detection accuracy of finite difference methods: In addition to the challenge of computational efficiency, most finite difference methods presented in this section are sensitive to the threshold parameter (ϵ) used to distinguish

between separable and nonseparable variables. Theoretically, a non-zero $\lambda = |\Delta^{(1)} - \Delta^{(2)}|$ signifies an interaction. However, computational roundoff errors can sometimes cause nonzero λ values even for separable variables. Therefore, when observing a nonzero value it is important to find whether it is caused by a genuine interaction or by computational errors. This clearly affects the choice of ϵ which has been investigated in several studies. gDG [156] normalizes the values which make it less sensitive and uses tighter threshold they call σ to detect interactions. This resulted in 100% accuracy almost all of the CEC'2010 benchmark suite. Cao et al. [165] extended gDG to decompose large-scale multiobjective problems. GDG [155] and EVIID [162] define ϵ to be a function of the objective function based on the rationale that the magnitude of the computational errors is a function of the objective function value: $\epsilon = \alpha \cdot \min\{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_k)\}$, where $\mathbf{x}_1 \ldots \mathbf{x}_k$ are k random samples of the search space, and α is a small user-defined parameter allegedly less sensitive to computational errors as compared to ϵ. Although GDG found the ideal decomposition for 18 out of 20 functions from the CEC'2010 LSGO benchmark suite, its performance deteriorates on the imbalanced functions of the CEC'2013 LSGO benchmark suite [166]. In FII [159] two different threshold values are used, one for detecting separable ($\lambda = |\Delta^{(1)} - \Delta^{(2)}| < \epsilon_1$), and another for the nonseparable variables ($\lambda = |\Delta^{(1)} - \Delta^{(2)}| > \epsilon_2$). Despite this suggestion, FII uses $\epsilon_1 = \epsilon_2 = 10^{-2}$ for the experiments.

Omidvar et al. [157] conducted a systematic error analysis of DG2 to place tight bounds on the roundoff errors. DG2 estimates the greatest lower bound $e_{\inf}$ and the least upper-bound $e_{\sup}$ of the computational errors. For each pair of variables, if the quantity $\lambda < e_{\inf}$, it is treated as genuine zero and the pair will be declared as separable; otherwise, if $\lambda > e_{\sup}$, it is treated as a genuine non-zero value and the pair will be declared as having interaction. For $\lambda \in [e_{\inf}, e_{\sup}]$, ϵ will be set to a weighted average of the two bounds based on the total number of genuine zero and nonzero detections. Unlike the previous finite difference methods, DG2 is parameter-free and calculates a different threshold value for each pair of the decision variables. On the CEC'2013 LSGO benchmark suite, DG2 outperformed CCVIL, DG, GDG, and XDG. Chen et al. [167] proposed global information-based adaptive threshold (GIAT) as an improved method for setting the threshold value based on $e_{\inf}$ and $e_{\sup}$. GIAT calculates these two quantities according to DG2, it then calculates the quantity $\zeta = \frac{(\lambda - e_{\inf}) f_s (\lambda - e_{\inf})}{\max\{|\Delta^{(1)}|, |\Delta^{(2)}|\}}$ for all pairs of variables similar to the way it is done by DG2. Finally, all ζ values are sorted and the two adjacent values with the largest difference are taken as the basis for calculating the threshold. GIAT uses this approach only on partially separable functions and sets the threshold value to the minimum of the two retrieved ζ values.

We close this section by reviewing two recent variants of RDG. RDG2 is the state-of-the-art decomposition algorithm that applies the error analysis mechanism of DG2 on the recursive mechanism of RDG to find the error bounds. This algorithm inherits the accuracy of DG2 and efficiency of RDG and outperforms both methods in grouping accuracy with a time complexity of $\mathcal{O}(n \log n)$ [168]. RDG3 [169], the winner of CEC'2019 Competition on Large-Scale Global Optimization[2], builds upon RDG2 and includes a mechanism to deal with problems with overlapping components[3].

2) Grouping Principles: This section revisits the explicit decomposition methods reviewed in the previous section and analyzes them based on their *grouping* mechanism rather than the *interaction detection* principles (Fig 5). The grouping principles can be classified into three major groups: automatic, semi-automatic, and k s-dimensional.

a) Automatic: The groups are either formed automatically in which case the number and the size of components are determined by the algorithm. This is usually done by processing the interaction information identified by the detection mechanisms outlined in §II-B1. For example, DG2 and GDG use the connected components algorithms on the interaction matrix of the function to form the groups. Other algorithms such as graphDG [156] use other graph partitioning techniques to form the groups. The automatic methods are predominantly based on differential grouping and monotonicity detection, which are among accurate interaction detection mechanisms (see Table S-I of the supplementary document). The variations in the grouping principles of these techniques mainly affect problems with overlapping components. Peng et al. [170] suggested a solution exchange scheme between components based on multimodal optimization to cope with grouping inaccuracies. Ren et al. [171] also proposed the bi-hierarchical cooperative coevolution which occasionally merge components to deal with decomposition inaccuracies. Overlapping problems are covered in part B of the survey.

b) Semi-automatic: Semi-automatic methods require the size or the number of components to be specified by the user. Cluster based methods such as the algorithm proposed by Fan et al. [172] use Fuzzy c-mean algorithm to form the groups which requires the number of components as input. Some studies [173, 174] use spectral clustering with differential grouping to take the degree of interaction into account. In statistical detection methods such as AVP2 [135] and 4CDE [136], a threshold or a set of intervals should be defined on the correlation coefficients to form the groups.

A special type of semi-automatic grouping is called *multilevel* [175, 176] in which the user specifies a list of potential component sizes for the algorithm to choose from. The algorithm often uses a probability distribution to choose a component size from the list and uniformly divide the n-dimensional problem into smaller components. The algorithm often adapts the parameters of the probability distribution based on the performance of the selected item in the course of optimization. Some multilevel algorithms however, use deterministic methods to gradually reduce the number of components during optimization [177].

c) k s-dimensional Components: These algorithms are the least informed and require both the number and the size of each component to form the groups. Detection principles

[2]For more information on LSGO competitions see part B of the survey .

[3]Problems with overlapping components are covered in part B of this survey

such as random grouping [118, 120, 178, 179], delta grouping [121], fitness difference partitioning [126, 127, 129, 130], and statistical methods [83, 134] are among such methods.

C. Advantages and disadvantages of explicit and implicit methods

Explicit and implicit methods each have their own merits. In dealing with partially additively separable functions, explicit methods are generally more efficient in finding the interaction structure using systematic perturbation methods than the implicit methods which relying on statistical information through random sampling. Although dimensionality is a major challenge to both types, the efficiency of implicit methods drop more sharply than explicit methods in higher dimensions. Accurate model estimation in implicit methods requires an exponentially increasing sample size with dimension [70], whereas the state-of-the-art decomposition algorithms require $\mathcal{O}(n \log n)$ to infer interactions [160]. When the interaction structure is more complicated and a "crisp" decomposition is not possible, implicit methods are generally more flexible in representing and exploiting such structures. For instance, when the underlying components of the objective function has shared decision variables (overlap), it is not clear how it should be decomposed into a set of disjoint groups (see §VI-A). Another advantage of implicit methods is that they build their model during optimization whereas explicit methods work offline and infer interactions prior to optimization. However, as was stated before, this flexibility comes at the cost of significant computational overhead to find a suitable model in the first place. For instance, BOAs use Bayesian networks which can represent any arbitrary interaction structure; however, discovering the "right" model is an NP-hard problem [46]. Another advantage of implicit methods, such as CMA-ES, is their rotational invariance property making them particularly suitable for solving nonseparable problems. Some studies attempted to combine decomposition methods with implicit methods to help EDAs scale better and bring the flexibility and the invariance property of the implicit methods to decompositions [80–83]

III. Hybridization and Memetic Algorithms

According to No Free Lunch theorem [180], no single search algorithm can uniformly outperform all other algorithms on all possible problems. This suggests that there are niche problem areas in which particular algorithms perform better than others. The aim of hybridization is to benefit from unique search capabilities of several algorithms to find high quality solutions to problems, which are better than the solutions obtained by the individual algorithms in isolation. More generally, the aim of hybridization in evolutionary algorithms is to [181]: 1) improve their performance (such as convergence speed); 2) improve the final solution quality obtained by such algorithms; and 3) incorporate such algorithms as part of a larger system. Some hybridization algorithms are generic in the sense that they can hybridize any number or combination of existing search algorithms. Ensemble strategies is a major allied topic concerned with the study of designing *stable* optimization algorithms by combining a set of "unstable and diverse" ones [182]. Hybrid local search and memetic algorithms play an important role in large-scale global optimization.

Hybrid local search algorithms

As distinct from memetic algorithms, these algorithms are solely based on local search with no explicit global search component. They often rely on an initial systematic initialization, such as orthogonal arrays [183], to attain a good coverage of the search space. Multiple trajectory search (MTS) [183] used three different local search methods employed based on the properties of the search space in the vicinity of existing candidate solutions. Before performing an extensive local search, MTS tests all three local search mechanism and picks the best mechanism that performs the best in that neighborhood. MTS has been tested on the CEC'2008 LSGO benchmark functions with up to 1000 dimensions. Gardeux et al. [184] also combined two line search algorithms, i.e. enhanced unidirectional search (EUS) and 3-2-3 line search algorithm, for large-scale global optimization. EUS searches along lines specified by a series of unit vectors not necessarily aligned with the coordinate system. Although variable interaction can have significant effect on the choice of direction vectors, this aspect has not been studied by the authors.

Memetic algorithms

Memetic algorithms [185] represent a special hybridization paradigm in which local search is applied to individuals within an explorative evolutionary framework to mimic the individual learning procedure. This dual-phase mechanism has the potential to balance between exploration and exploitation forces, which are inspired by Darwinian evolution and the effect of individual learning considered in Baldwinian/Lamarckian evolution, respectively. Memetic algorithms have gained popularity in large-scale global optimization, some of which ranked first in IEEE CEC Competition on Large-Scale Global Optimization[4]. Memetic algorithms have also been applied to discrete optimization [186], boolean satisfiability problems [187] and a range of application areas such as large-scale hybrid flow shop problems [188], large-scale capacitated arc routing problems [189, 190], and big data optimization problems [191].

Major design considerations in memetic algorithms are [192]: 1) frequency of applying local search; 2) choice of solutions participating in local search (individual learning); 3) search intensity, i.e., the duration of applying local search; and 4) choice of local search algorithms. In addition to the above, the algorithms using a repertoire of local search procedures need to devise a policy to choose from the available local search operators. In what follows, we review the relevant memetic and other hybrid algorithms used for large-scale global optimization in reference to the above design considerations.

a) Local search frequency: The most commonly used way of controlling the frequency of applying local search is

[4]For more information on competition results refer to part B of the survey.

to run at regular interval as controlled by a user-defined parameter [193–195]. An extreme case for this approach is to run the local search procedure at every iteration [183, 196–198]. In some cases if the local search process is marked as stagnant, it will not be invoked [196]. MTS, which uses multiple local search operators, runs all its operators at the beginning of every cycle and picks the best performing operator for the rest of that cycle. In some algorithms, instead of resorting to running the local search at fixed regular intervals, the local search operators are applied such that the ratio between the local and global search is fixed [199–201]. In some algorithms such as MA-SW-Chains [199, 200], search frequency and intensity (covered later in this section) are linked through a fixed local/global search ratio. In other words, once the global search ratio is fixed, specifying search frequency or intensity determines the other. Some other approaches decide the invocation of local search probabilistically using a pre-specified distribution whose parameters are set and/or adapted based on its success/failure rate [202, 203].

b) Choice of solutions: The choice of solutions to participate in the local search process can be random, performance-based, complete (i.e., all solutions participate) [198], or any combination of the three. For example, the algorithm proposed by Zhao et al. [203] chooses 5 random solutions, the best solution, and the top 25%. The performance-based methods either apply local search to the best solution [194–197, 201, 202], top n solutions [183], top $p\%$ solutions [193], or those improved more than a predefined threshold in previous iterations [199, 200]. Some methods also emphasize the solutions not selected before [199, 200]. Zhao et al. [203] proposed to use the niching algorithm Clearing proposed by Pétrowski [204] to choose the solutions to undergo local search. They start from several solutions which progressively reduces to one, with the aim of controlling the exploration/exploitation balance.

c) Search intensity: Search intensity is defined as the duration in which the local search is active. The simplest way of specifying the search intensity is a fixed-budget policy, i.e, to run the local search operator for a fixed and predetermined number of function evaluations [193, 196, 201, 202]. MTS [183] uses a greedy approach and runs the local search algorithm until no improvement is observed. Other algorithms using MTS-based local optimizers also use a similar strategy [194, 195]. As was discussed previously in the context of search frequency, some algorithms link search frequency and intensity by forcing a fixed local/global search ratio [199, 200]. A more sophisticated way of specifying search intensity is to do so adaptively during the course of optimization. Liu and Li [197] determine the search intensity based on the success/failure rate of the local search procedure. Bolufé-Röhler et al. [198] simply double the search intensity every time the local search procedure is invoked. Zhao et al. [203] run the global and local search procedures once at the beginning of each cycle, and the search intensity for each case is specified proportional to their success/failure rates.

d) The local search procedure: A wide range of global and local search algorithms have been hybridized for solving large-scale global optimization algorithms. Table I summarizes various hybridizations proposed in the literature. In the table 'G' denotes a global search mechanism, 'L' denotes a local search procedure, the combination of which indicate a memetic algorithm, whereas 'H' denotes a generic hybridization of a set of local or global search procedures. The table shows that differential evolution (DE) [220] is the most widely used global search mechanism followed by particle swarm optimization (PSO) [221]. A wide range of local search operators are also used that can be categorized as random search, line search, and coordinate descent. Algorithms from all three categories are commonly used in various memetic or hybrid frameworks; however, the coordinate decent procedure of MTS [183], known as MTS-LS1, and the Solis and Wets' [222] random search algorithms are the most popular operators.

In most cases, a single local search procedure is used within a global exploratory process. However, in some cases more than two operators, local or global, are used simultaneously, the application of which needs to be coordinated by the hybrid framework. For instance, an algorithm proposed by Fister et al. [201] probabilistically selects between random walk and Nelder-Mead [223] based on an exponential distribution whose parameter is adapted according to a measure of population diversity. Another algorithm by Sabar et al. [191] adaptively selects between Rosenbrock's [205] and Powell's [206] search algorithms by testing their effectiveness using statistical hypothesis testing methods during the course of optimization. MTS coordinates between three local search operators by running them on m solutions at the beginning of each cycle and uses the best performing operator for the rest of the cycle until it becomes stagnant. There are also some algorithms [125, 213] which employ memetic algorithms in a cooperative coevolution framework. These algorithms decompose the problem into separable and nonseparable components and employ different local search operators suitable for the separable and the nonseparable components. Further details of coevolutionary memetic algorithms are given later in this section.

Multiple offspring (MOS) framework [224] is an abstraction layer on top of the reproductive operators of existing evolutionary algorithms, which systematizes the coordination of several search operators. MOS employs a repertoire of evolutionary operators and applies them based on their performance over the course of optimization in order to achieve a higher long-term performance. In the context of large-scale optimization, several evolutionary operators have been hybridized using MOS framework [208–210], which are summarized in Table I. The experimental results on a wide range of benchmark functions with up to 1000 dimensions showed the scalability of MOS framework to high-dimensional problems, making it the first-ranked algorithm in CEC'2013 and CEC'2015 competition on large-scale global optimization. The reader is referred to part B of the survey for a discussion on LSGO competitions and more recent results.

e) Parallel vs Sequential Hybrids: Memetic algorithms and other hybrid methods can be implemented in a parallel paradigm or a sequential one. In the context of large-scale global optimization, parallel hybrids are limited [16, 214], with most algorithms following a sequential paradigm. Wang

TABLE I: Summary of common algorithms used in hybridization frameworks. The algorithms are classified into: Global Search, Random Search, Line Search, Coordinate Descent, and other derivative-free methods such as Nelder-Mead, CMA-ES, or Tabu Search. The character G denotes that the algorithm is used as a global search operator, L denotes a local search operator, and H denotes a generic hybridization of a set of local or global search operators.

Citation	Differential Evolution	Particle Swarm Opt.	Genetic Algorithm	Ant Colony Opt.	Artificial Bee Colony	Migrating Birds Opt.	Coral Reefs Opt.	Min Population Search	Solis and Wets	Random Walk	Rosenbrock[205]	Powell[206]	Quasi-Newton	Line Search (Misc)	MTS LS1	MTS LS2	MTS LS3	Coord Descent (Misc)	Nelder-Mead	CMA-ES/EDA	Tabu Search
	Global Search							Random		Line Search				Coordinate Descent				Other			
Bolufé-Röhler et al. [198]								G												L	
Cao et al. [207]	G								L												
de Oca et al. [196]		G										L									
Fister et al. [201]					G					L									L		
Gardeux et al. [184]														L/L							
LaTorre et al. [208]	G														L						
LaTorre et al. [209]									H						H						
LaTorre et al. [210]			H						H						H*						
Li and Pan [188]					G																L
Liu and Li [197]	G								L												
Liu et al. [125]	G												L					L			
Molina et al. [211]															H	H			H		
Molina et al. [199]			G						L												
Molina et al. [200]			G						L												
Molina and Herrera [202]	G												L								
Olguin-Carbajal et al. [194]	G														L						
Sabar et al. [191]	G										L	L									
Salcedo-Sanz et al. [195]							G								L						
Segredo et al. [212]	H					H															
Seren [186]		G																			
Sun et al. [213]	G										L				L						
Tang et al. [214]																					
Tseng and Chen [183]															H	H	H				
Vitorino et al. [215]		H			H																
Wang et al. [216]	H																			H	
Yang and Sato [217]	H	H																			
Ye et al. [218]	H/H																				
Zhao et al. [193]		G											L								
Zhao et al. [203]	G																	L*			
Deng et al. [219]		H	H	H																	
Total	14	6	4	1	3	1	1	1	6	1	2	2	3	2	8	2	1	2	2	2	1

et al. [216] suggest that parallel hybrids may not be effective in utilizing the benefits of various search algorithms due to disparities in their convergence speed and diversity maintenance. Molina et al. [225] proposed the idea of *local search chains* in which also emphasizes a sequential paradigm. The aim of these search chains is to perform an intensive local search during the course of optimization. The term *chain* alludes to the fact that a local search operator can be applied in succession, and each invocation can resume the search process from where it stopped in its previous invocation. Hence, forming a chain of local searches has the capacity to better exploit the properties of the landscape and focus on more promising regions. The idea of local search chains was first used with CMA-ES [68] as the local search operator to form the MA-CMA-Chains algorithm [225]. The computational cost of CMA-ES makes it prohibitive for large-scale optimization. Therefore, Molina et al. [199] employed the Solis Wets' [222] algorithm as the local search operator and proposed MA-SW-Chains. This algorithm, ranked first in the CEC'2010 Competition on Large-Scale Optimization, showed better performance relative to other algorithms on the CEC'2010 large-scale benchmark problems [122]. Later, a variant of MA-SW-Chains, called MA-SSW-Chains was developed in which the local search was only applied to a random subset of the decision variables [200].

f) Cooperative Coevolution and Memetic Algorithms: Decomposition methods (see §II-B) and memetic algorithms are the two most widely used approaches to large-scale global optimization with algorithms from both categories ranked first in large-scale global optimization competitions [169, 226, 227]. To benefit from the advantages of both approaches, some authors suggest the use of memetic algorithms [125, 126, 191, 207, 213] or other hybrids [218] as component optimizers in a cooperative coevolution framework. The general approach is to decompose the problem into a set of lower dimensional subproblems using the methods described in §II-B, and optimize each component using a global search algorithm followed by an episode of local search. Cao et al. [207] proposed to use SaNSDE [228] followed by Solis and Wets' [222] on each component and adjust their search intensity/frequency according to their performance. Sun et al. [213] also used SaNSDE as the global search operator followed by dedicated local search procedures for the separable and nonseparable components. Sabar et al. [191] use two local search operators (i.e., Rosenbrock's [205] and Powell's [206]) in conjunction with DE as the global search algorithm. Liu et al. [125]

proposed to use coordinate descent and Quasi-Newton local search algorithms on separable and nonseparable components respectively, followed by a round of DE to further improve the population.

The MLSHADE-SPA algorithm Hadi et al. [229], runner-up of the IEEE CEC'2018 large-scale competition, takes a different approach to combining memetic and coevolutionary search. The algorithm divides the entire search process into several rounds where each round is comprised of an initial coevolutionary search phased followed by a local search phase. In the coevolutionary phase, the problem is decomposed into three equally-sized components each of which is optimized using a different DE-based algorithm. The coevolutionary phase is followed by local search where the best found solution is improved using a modified MTS-LS1 algorithm [183].

IV. Concluding Remarks

In this part of the series, we covered two major approaches to large-scale global optimization: 1) Algorithms which exploit problem structure in the form of variable interaction, and 2) Hybrid algorithms, most notably memetic algorithms and local search.

Exploiting problem structure and grey-box optimization has shown to be effective ways of solving large-scale problems (§II). These structural information can be used in the form of explicit decomposition or implicitly through model building. The challenge of explicit methods is the cost of offline variable interaction learning, which requires objective function evaluations and causes an overhead on the overall optimization cost. Another issue is that a crisp decomposition is sometimes impractical due to various forms of couplings caused by the existence of multiple objectives, overlapping components (shared variables among subfunctions), or coupling through constraints. Implicit methods also suffer from the accuracy of capturing problem structure, especially when the problem size grows in size. Finding more efficient and effective ways of exploiting structural information, such as overlap, can have a significant impact on improving the scalability of optimization algorithms.

Hybrid methods and memetic algorithms in particular, use the available computational budget in a more economical way and gain competitive advantage by means of extensive local search. It is not clear how these algorithms may perform in finding global optimum under more relaxed budget constraints. Their dimension-wise local search procedures are generally blind to variable interactions making them better suited for separable functions. In memetic frameworks the design considerations, such as the frequency of local search or the choice of the local search procedure, are ad hoc and require extensive experimentation. Designing generic frameworks capable of finding the optimal search intensity and the choice of local search procedures can significantly improve the performance of these algorithms and also make them readily available to practitioners in other fields.

In the next part of this survey series, we cover several other approaches to large-scale global optimization and also look at several important problem areas such as multi-objective optimization and constraint handling. The next part also touches upon two major issues pertaining to the future of the field: 1) Pitfalls and challenges that hinder the progress of the field, and 2) The pressing open questions and potential areas of future research.

Acknowledgements

This work was partially supported by an ARC (Australian Research Council) Discovery Grant (DP180101170), Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (Grant No. 2017ZT07X386), and the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008).

References

[1] R. E. Bellman, *Adaptive control processes: a guided tour*. Princeton university press, 1961.

[2] P. Drineas and M. W. Mahoney, "RandNLA: randomized numerical linear algebra," *Communications of the ACM*, vol. 59, no. 6, pp. 80–90, 2016.

[3] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[4] N. Gould, D. Orban, and P. Toint, "Numerical methods for large-scale nonlinear optimization," *Acta Numerica*, vol. 14, pp. 299–361, 2005.

[5] N. Hollister and A. Wood, "The tallest 20 in 2020: Entering the era of the megatall," *Press Release. Chicago*, 2011.

[6] M. Gilbert, L. He, H. Lu, A. Tyas, H. E. Fairclough, J. Gondzio, and A. G. Weldeyesus, "Layout optimization of large-scale trusses and frames," in *Proceedings of IASS Annual Symposia*, vol. 2018, no. 19. International Association for Shell and Spatial Structures (IASS), 2018, pp. 1–8.

[7] Z.-H. Zhou, N. V. Chawla, Y. Jin, and G. J. Williams, "Big data opportunities and challenges: Discussions from data analytics perspectives," *IEEE Computational Intelligence Magazine*, vol. 9, no. 4, pp. 62–74, 2014.

[8] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[9] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*. Omnipress, 2011, pp. 265–272.

[10] G. N. Vanderplaats, "Very large scale optimization," National Aeronautics and Space Administration (NASA), Langley Research Center, Colorado, US, Tech. Rep. NASA/CR-2002-211768, 2002.

[11] S. Mahdavi, M. E. Shiri, and S. Rahnamayan, "Metaheuristics in large-scale global continues optimization: A survey," *Information Sciences*, vol. 295, pp. 407–428, Feb. 2015.

[12] G. Morse and K. O. Stanley, "Simple evolutionary optimization can rival stochastic gradient descent in neural networks," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*. ACM, 2016, pp. 477–484.

[13] K. Deb and C. Myburgh, "A population-based fast algorithm for a billion-dimensional resource allocation problem with integer variables," *European Journal of Operational Research*, vol. 261, no. 2, pp. 460–474, 2017.

[14] K. Sastry, D. E. Goldberg, and X. Llora, "Towards billion-bit optimization via a parallel estimation of distribution algorithm," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '07. New York, NY, USA: Association for Computing Machinery, Jul. 2007, pp. 577–584.

[15] A. Cano and C. García-Martínez, "100 million dimensions large-scale global optimization using distributed gpu computing," in *IEEE Congress on Evolutionary Computation*. IEEE, 2016, pp. 3566–3573.

[16] A. Cano, C. García-Martínez, and S. Ventura, "Extremely high-dimensional optimization with mapreduce: scaling functions and algorithm," *Information Sciences*, vol. 415, pp. 110–127, 2017.

[17] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," *arXiv preprint arXiv:1712.06567*, 2018.

[18] J.-R. Jian, Z.-H. Zhan, and J. Zhang, "Large-scale evolutionary optimization: A survey and experimental comparative study," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 3, pp. 729–745, Mar. 2020.

[19] D. Thierens, "Scalability problems of simple genetic algorithms," *Evolutionary Computation*, vol. 7, no. 4, pp. 331–352, 1999.

[20] D. Molina, M. Lozano, and F. Herrera, "Memetic algorithm with local search chaining for continuous optimization problems: A scalability test," in *2009 Ninth International Conference on Intelligent Systems Design and Applications*. IEEE, 2009, pp. 1068–1073.

[21] F. Caraffini, F. Neri, and L. Picinali, "An analysis on separability for memetic computing automatic design," *Information Sciences*, vol. 265, pp. 1–22, 2014.

[22] Y. Sun, M. Kirley, and S. K. Halgamuge, "Quantifying variable interactions in continuous optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 2, pp. 249–264, 2017.

[23] N. Hansen, "Adaptive encoding: How to render search coordinate system invariant," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2008, pp. 205–214.

[24] T. Hogg, "Exploiting problem structure as a search heuristic," *International Journal of Modern Physics C*, vol. 9, no. 01, pp. 13–29, 1998.

[25] U. Aickelin and K. A. Dowsland, "Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem," *Journal of scheduling*, vol. 3, no. 3, pp. 139–153, 2000.

[26] C. J. Price and P. L. Toint, "Exploiting problem structure in pattern search methods for unconstrained optimization," *Optimisation Methods and Software*, vol. 21, no. 3, pp. 479–491, 2006.

[27] N. Ho-Nguyen and F. Kılınç-Karzan, "Exploiting problem structure in optimization under uncertainty via online convex optimization," *Mathematical Programming*, pp. 1–35, 2018.

[28] R. Santana, "Gray-box optimization and factorized distribution algorithms: where two worlds collide," *arXiv preprint arXiv:1707.03093*, 2017.

[29] J. Holland, *Hidden Order: How Adaptation Builds Complexity*. Perseus, 1995.

[30] G. R. Harik and D. E. Goldberg, "Learning linkage." in *Foundations of Genetic Algorithms*, vol. 4, 1996, pp. 247–262.

[31] Y. Chen, T.-L. Yu, K. Sastry, and D. E. Goldberg, "A survey of linkage learning techniques in genetic and evolutionary algorithms," University of Illinois at Urbana-Champaign, Tech. Rep. 2007014, 2007.

[32] D. Goldberg, B. Korb, and K. Deb, "Messy genetic algorithms: Motivation, analysis, and first results," *Complex Systems*, vol. 3, no. 5, pp. 493–530, 1989.

[33] D. E. Goldberg, K. Deb, and D. Thierens, "Toward a better understanding of mixing in genetic algorithms," *Journal of the Society of Instrument and Control Engineers*, vol. 32, no. 1, pp. 10–16, 1993.

[34] D. Thierens and D. E. Goldberg, "Mixing in genetic algorithms," *Urbana*, vol. 51, p. 61801, 1993.

[35] R. Salomon, "Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms," *BioSystems*, vol. 39, no. 3, pp. 263–278, 1996.

[36] D. R. Hains, L. D. Whitley, and A. E. Howe, "Revisiting the big valley search space structure in the tsp," *Journal of the Operational Research Society*, vol. 62, no. 2, pp. 305–312, 2011.

[37] D. Whitley, D. Hains, and A. Howe, "Tunneling between optima: partition crossover for the traveling salesman problem," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. ACM, 2009, pp. 915–922.

[38] J. G. Monroe, Z. A. Allen, P. Tanger, J. L. Mullen, J. T. Lovell, B. T. Moyers, D. Whitley, and J. K. McKay, "Tspmap, a tool making use of traveling salesperson problem solvers in the efficient and accurate construction of high-density genetic linkage maps," *BioData mining*, vol. 10, no. 1, p. 38, 2017.

[39] R. Tintos, D. Whitley, and F. Chicano, "Partition crossover for pseudo-boolean optimization," in *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*. ACM, 2015, pp. 137–149.

[40] G. Wu, W. Pedrycz, P. N. Suganthan, and R. Mallipeddi, "A variable reduction strategy for evolutionary algorithms handling equality constraints," *Applied Soft Computing*, vol. 37, pp. 774–786, Dec. 2015.

[41] G. Wu, W. Pedrycz, P. N. Suganthan, and H. Li, "Using variable reduction strategy to accelerate evolutionary optimization," *Applied Soft Computing*, vol. 61, pp. 283–293, Dec. 2017.

[42] A. H. Gandomi, K. Deb, R. C. Averill, S. Rahnamayan, and M. N. Omidvar, "Using semi-independent variables to enhance optimization search," *Expert Systems with Applications*, vol. 120, pp. 279–297, 2019.

[43] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Proc. of International Conference on Parallel Problem Solving from Nature*, vol. 2, 1994, pp. 249–257.

[44] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, Tech. Rep., 1994.

[45] M. Pelikan, D. E. Goldberg, and F. G. Lobo, "A survey of optimization by building and using probabilistic models." *Comp. Opt. and Appl.*, vol. 21, no. 1, pp. 5–20, 2002.

[46] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The bayesian optimization algorithm," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*. Morgan Kaufmann Publishers Inc., 1999, pp. 525–532.

[47] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: University of Michigan Press, 1975.

[48] D. E. Goldberg, R. Lingle *et al.*, "Alleles, loci, and the traveling salesman problem," in *Proceedings of an international conference on genetic algorithms and their applications*, vol. 154. Lawrence Erlbaum, Hillsdale, NJ, 1985, pp. 154–159.

[49] D. E. Goldberg and C. L. Bridges, "An analysis of a reordering operator on a ga-hard problem," *Biological Cybernetics*, vol. 62, no. 5, pp. 397–405, 1990.

[50] D. E. Goldberg, K. Deb, H. Kargupta, and G. Harik, "Rapid, accurate optimization of difficult problems using fast messy genetic algorithms," in *Proc. of International Conference on Genetic Algorithms*, S. Forrest, Ed. San Francisco, CA: Morgan Kaufmann, 1993, pp. 56–64.

[51] H. Kargupta, "The gene expression messy genetic algorithm," in *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*. IEEE, 1996, pp. 814–819.

[52] G. R. Harik, "Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms," Ph.D. dissertation, University of Michigan, Ann Arbor, MI, 1997.

[53] J. Smith and T. C. Fogarty, "An adaptive poly-parental recombination strategy," in *Selected Papers from AISB Workshop on Evolutionary Computing*. London, UK: Springer-Verlag, 1995, pp. 48–61.

[54] H. Mühlenbein and G. Paass, "From recombination of genes to the estimation of distributions i. binary parameters," in *Proc. of International Conference on Parallel Problem Solving from Nature*. London, UK: Springer-Verlag, 1996, pp. 178–187.

[55] G. Harik, F. Lobo, and D. Goldberg, "The compact genetic algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 287–297, November 1999.

[56] M. Pelikan, D. E. Goldberg, and S. Tsutsui, "Combining the strengths of bayesian optimization algorithm and adaptive evolution strategies," in *Proc. of Genetic and Evolutionary Computation Conference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 512–519.

[57] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[58] T.-L. Yu, D. E. Goldberg, K. Sastry, C. F. Lima, and M. Pelikan, "Dependency structure matrix, genetic algorithms, and effective recombination," *Evolutionary computation*, vol. 17, no. 4, pp. 595–626, 2009.

[59] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, N. De Freitas *et al.*, "Bayesian optimization in high dimensions via random embeddings." in *IJCAI*, 2013, pp. 1778–1784.

[60] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. de Feitas, "Bayesian optimization in a billion dimensions via random embeddings," *Journal of Artificial Intelligence Research*, vol. 55, pp. 361–387, 2016.

[61] T. N. Hoang, Q. M. Hoang, R. Ouyang, and K. H. Low, "Decentralized high-dimensional bayesian optimization with factor graphs," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[62] P. Rolland, J. Scarlett, I. Bogunovic, and V. Cevher, "High-dimensional bayesian optimization via additive models with overlapping groups," *arXiv preprint arXiv:1802.07028*, 2018.

[63] S. Rana, C. Li, S. Gupta, V. Nguyen, and S. Venkatesh, "High dimensional bayesian optimization with elastic gaussian process," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2883–2891.

[64] C.-L. Li, K. Kandasamy, B. Póczos, and J. Schneider, "High dimensional bayesian optimization via restricted projection pursuit models," in *Artificial Intelligence and Statistics*, 2016, pp. 884–892.

[65] C. Li, S. Gupta, S. Rana, V. Nguyen, S. Venkatesh, and A. Shilton, "High dimensional bayesian optimization using dropout," *arXiv preprint arXiv:1802.05400*, 2018.

[66] Z. Wang, C. Li, S. Jegelka, and P. Kohli, "Batched high-dimensional

bayesian optimization via structural kernel learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3656–3664.

[67] K. Kandasamy, J. Schneider, and B. Póczos, "High dimensional bayesian optimisation and bandits via additive models," in *International Conference on Machine Learning*, 2015, pp. 295–304.

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

[69] K. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[70] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics Springer, Berlin, 2001, vol. 1.

[71] R. Vershynin, "Introduction to the non-asymptotic analysis of random matrices," *arXiv preprint arXiv:1011.3027*, 2010.

[72] W. Dong and X. Yao, "Unified eigen analysis on multivariate gaussian based estimation of distribution algorithms," *Information Sciences*, vol. 178, no. 15, pp. 3000–3023, 2008.

[73] Y. Wang and B. Li, "A restart univariate estimation of distribution algorithm: sampling under mixed gaussian and lévy probability distribution," in *IEEE Congress on Evolutionary Computation*. IEEE, 2008, pp. 3917–3924.

[74] ——, "A self-adaptive mixed distribution based uni-variate estimation of distribution algorithm for large scale global optimization," in *Nature-Inspired Algorithms for Optimisation*. Springer, 2009, pp. 171–198.

[75] H. Mühlenbein and T. Mahnig, "Convergence theory and applications of the factorized distribution algorithm," *Journal of Computing and Information Theory*, vol. 7, no. 1, pp. 19–32, 1999.

[76] P. Larrañaga and J. Lozano, *Estimation of Distribution Algorithms: A new tool for evolutionary computation*. Kluwer Academic Pub, 2002.

[77] C. Echegoyen, Q. Zhang, A. Mendiburu, R. Santana, J. Lozano *et al.*, "On the limits of effectiveness in estimation of distribution algorithms," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE, 2011, pp. 1573–1580.

[78] W. Dong, T. Chen, P. Tino, and X. Yao, "Scaling up estimation of distribution algorithms for continuous optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 6, pp. 797–822, 2013.

[79] Q. Xu, M. L. Sanyang, and A. Kab
'an, "Large scale continuous eda using mutual information," in *IEEE Congress on Evolutionary Computation*. IEEE, 2016, pp. 3718–3725.

[80] Y. Wang and B. Li, "Two-stage based ensemble optimization for large-scale global optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.

[81] Y. Wang, J. Huang, W. S. Dong, J. C. Yan, C. H. Tian, M. Li, and W. T. Mo, "Two-stage based ensemble optimization framework for large-scale global optimization," *European Journal of Operational Research*, vol. 228, no. 2, pp. 308–320, 2013.

[82] K. Zhang and B. Li, "Cooperative coevolution with global search for large scale global optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–7.

[83] J. Liu and K. Tang, "Scaling up covariance matrix adaptation evolution strategy using cooperative coevolution," in *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 2013, pp. 350–357.

[84] A. Kabán, J. Bootkrajang, and R. J. Durrant, "Toward large-scale continuous eda: A random matrix theory perspective," in *Genetic and Evolutionary Computation Conference*. ACM, 2013, pp. 255–291.

[85] A. Kabán, J. Bootkrajang, and R. J. Durrant, "Towards large scale continuous eda: A random matrix theory perspective," *Evolutionary Computation*, May 2015.

[86] S. Dasgupta, "Learning mixtures of gaussians," in *Foundations of Computer Science, 1999. 40th Annual Symposium on*. IEEE, 1999, pp. 634–644.

[87] P. Diaconis and D. Freedman, "Asymptotics of graphical projection pursuit," *The annals of statistics*, pp. 793–815, 1984.

[88] W. Dong, Y. Wang, and M. Zhou, "A latent space-based estimation of distribution algorithm for large-scale global optimization," *Soft Computing*, vol. 23, no. 13, pp. 4593–4615, Jul. 2019.

[89] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[90] C.-Y. Lee and X. Yao, "Evolutionary programming using mutations based on the lévy probability distribution," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 1–13, 2004.

[91] T. Schaul, T. Glasmachers, and J. Schmidhuber, "High dimensions and heavy tails for natural evolution strategies," in *Proc. of the 13th annual Conference on Genetic and Evolutionary Computation*. ACM, 2011, pp. 845–852.

[92] M. L. Sanyang, H. Muehlbrandt, and A. Kaban, "Two approaches of using heavy tails in high dimensional eda," in *IEEE International Conference on Data Mining Workshop*. IEEE, 2014, pp. 653–660.

[93] M. L. Sanyang and A. Kabán, "Heavy tails with parameter adaptation in random projection based continuous eda," in *IEEE Congress on Evolutionary Computation*. IEEE, 2015, pp. 2074–2081.

[94] P. Pošík, "Bbob-benchmarking a simple estimation of distribution algorithm with cauchy distribution," in *Proc. of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. ACM, 2009, pp. 2309–2314.

[95] M. L. Sanyang and A. Kaban, "Multivariate cauchy eda optimisation," in *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 2014, pp. 449–456.

[96] M. L. Sanyang, R. J. Durrant, and A. Kabán, "How effective is cauchy-eda in high dimensions?" in *IEEE Congress on Evolutionary Computation*. IEEE, 2016, pp. 3409–3416.

[97] N. Hansen, F. Gemperle, A. Auger, and P. Koumoutsakos, "When do heavy-tail distributions help?" in *Parallel Problem Solving from Nature*. Springer, 2006, pp. 62–71.

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

[99] R. Ros and N. Hansen, "A simple modification in cma-es achieving linear time and space complexity," in *Parallel Problem Solving from Nature*. Springer, 2008, pp. 296–305.

[100] C. Igel, T. Suttorp, and N. Hansen, "A computational efficient covariance matrix update and a (1+1)-cma for evolution strategies," in *Proc. of Conference on Genetic and Evolutionary Computation*. ACM, 2006, pp. 453–460.

[101] J. Poland and A. Zell, "Main vector adaptation: A cma variant with linear time and space complexity," in *Proc. of Genetic and Evolutionary Computation Conference*. Morgan Kaufmann Publishers Inc., 2001, pp. 1050–1055.

[102] J. N. Knight and M. Lunacek, "Reducing the space-time complexity of the cma-es," in *Genetic and Evolutionary Computation Conference*. ACM, 2007, pp. 658–665.

[103] Y. Sun, T. Schaul, F. Gomez, and J. Schmidhuber, "A linear time natural evolution strategy for non-separable functions," in *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*. ACM, 2013, pp. 61–62.

[104] I. Loshchilov, "A computationally efficient limited memory CMA-ES for large scale optimization," in *Genetic and Evolutionary Computation Conference*. ACM, 2014, pp. 397–404.

[105] ——, "Lm-cma: An alternative to L-BFGS for large-scale black box optimization," *Evolutionary computation*, vol. 25, no. 1, pp. 143–171, 2017.

[106] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, pp. 503–528, 1989.

[107] Z. Li, Q. Zhang, X. Lin, and H.-L. Zhen, "Fast Covariance Matrix Adaptation for Large-Scale Black-Box Optimization," *IEEE Transactions on Cybernetics*, vol. 50, no. 5, pp. 2073–2083, May 2020.

[108] H.-G. Beyer and B. Sendhoff, "Simplify Your Covariance Matrix Adaptation Evolution Strategy," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 746–759, Oct. 2017.

[109] I. Loshchilov, T. Glasmachers, and H.-G. Beyer, "Large Scale Black-Box Optimization by Limited-Memory Matrix Adaptation," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 353–358, Apr. 2019.

[110] Z. Li and Q. Zhang, "A simple yet efficient evolution strategy for large scale black-box optimization," *IEEE Transactions on Evolutionary Computation*, 2017.

[111] X. He, Z. Zheng, and Y. Zhou, "MMES: Mixture Model-Based Evolution Strategy for Large-Scale Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 2, pp. 320–333, Apr. 2021.

[112] D. Thierens, "The linkage tree genetic algorithm," in *Parallel Problem Solving from Nature*. Springer, 2010, pp. 264–273.

[113] M. Tsuji, M. Munetomo, and K. Akama, "Linkage identification by fitness difference clustering," *Evolutionary Computation*, vol. 14, no. 4, pp. 383–409, 2006.

[114] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[115] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary

programming with cooperative coevolution," in *Proc. of IEEE Congress on Evolutionary Computation*, 2001, pp. 1101–1108.

[116] Y. Shi, H. Teng, , and Z. Li, "Cooperative co-evolutionary differential evolution for function optimization," in *Proc. of International Conference on Natural Computation*, 2005, pp. 1080–1088.

[117] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization 11 (4)*, pp. 341–359, 1995.

[118] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178, no. 15, pp. 2985–2999, 2008.

[119] X. Li and X. Yao, "Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms," in *IEEE Congress on Evolutionary Computation*. IEEE, 2009, pp. 1546–1553.

[120] M. N. Omidvar, X. Li, Z. Yang, and X. Yao, "Cooperative co-evolution for large scale optimization through more frequent random grouping," in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.

[121] M. N. Omidvar, X. Li, and X. Yao, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.

[122] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, "Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization," Nature Inspired Computation and Applications Laboratory, USTC, China, Tech. Rep., 2009, http://nical.ustc.edu.cn/cec10ss.php.

[123] R. Salomon, "Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions - a survey of some theoretical and practical aspects of genetic algorithms," *BioSystems*, vol. 39, pp. 263–278, 1995.

[124] H. Ge, M. Zhao, Y. Hou, Z. Kai, L. Sun, G. Tan, Q. Zhang, and C. L. Philip Chen, "Bi-space Interactive Cooperative Coevolutionary algorithm for large scale black-box optimization," *Applied Soft Computing*, vol. 97, p. 106798, Dec. 2020.

[125] H. Liu, Y. Wang, L. Liu, X.-Z. Gao, and Y.-m. Cheung, "A new grouping strategy-based hybrid algorithm for large scale global optimization problems," in *Genetic and Evolutionary Computation Conference Companion*. ACM, 2017, pp. 171–172.

[126] E. Sayed, D. Essam, and R. Sarker, "Dependency identification technique for large scale optimization problems," in *IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–8.

[127] A. E. Aguilar-Justo and E. Mezura-Montes, "Towards an improvement of variable interaction identification for large-scale constrained problems," in *IEEE Congress on Evolutionary Computation*. IEEE, 2016, pp. 4167–4174.

[128] A. E. Aguilar-Justo, E. Mezura-Montes, S. M. Elsayed, and R. A. Sarker, "Decomposition of large-scale constrained problems using a genetic-based search," in *2016 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*. IEEE, 2016, pp. 1–6.

[129] E. Sayed, D. Essam, R. Sarker, and S. Elsayed, "Decomposition-based evolutionary algorithm for large scale constrained problems," *Information Sciences*, vol. 316, pp. 457–486, 2015.

[130] G. Dai, X. Chen, L. Chen, M. Wang, and L. Peng, "Cooperative coevolution with dependency identification grouping for large scale global optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2016, pp. 5201–5208.

[131] A. Tiwari, R. Roy, G. Jared, and O. Munaux, "Interaction and multi-objective optimisation," in *Genetic and Evolutionary Computation Conference*. Morgan Kaufmann Publishers Inc., 2001, pp. 671–678.

[132] A. Tiwari and R. Roy, "Variable dependence interaction and multi-objective optimisation," in *Genetic and Evolutionary Computation Conference*. Morgan Kaufmann Publishers Inc., 2002, pp. 602–609.

[133] R. Roy and A. Tiwari, "Generalised regression GA for handling inseparable function interaction: Algorithm and applications," in *Parallel Problem Solving from Nature*. Springer, 2002, pp. 452–461.

[134] T. Ray and X. Yao, "A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning," in *IEEE Congress on Evolutionary Computation*. IEEE, 2009, pp. 983–989.

[135] H. K. Singh and T. Ray, "Divide and conquer in coevolution: A difficult balancing act," in *Agent-Based Evolutionary Search*. Springer, 2010, pp. 117–138.

[136] Y. Rojas and R. Landa, "Towards the use of statistical information and differential evolution for large scale global optimization," in *International Conference on Electrical Engineering Computing Science and Automatic Control*. IEEE, 2011, pp. 1–6.

[137] Y. Wang, B. Li, and X. Lai, "Variance priority based cooperative coevolution differential evolution for large scale global optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2009, pp. 1232–1239.

[138] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti, "Detecting novel associations in large data sets," *science*, vol. 334, no. 6062, pp. 1518–1524, 2011.

[139] S. Mahdavi, M. E. Shiri, and S. Rahnamayan, "Cooperative co-evolution with a new decomposition method for large-scale optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2014, pp. 1285–1292.

[140] I. M. Sobol, "Sensitivity estimates for nonlinear mathematical models," *Mathematical modelling and computational experiments*, vol. 1, no. 4, pp. 407–414, 1993.

[141] S. Shan and G. G. Wang, "Metamodeling for high dimensional simulation-based design problems," *Journal of Mechanical Design*, vol. 132, no. 5, p. 051009, 2010.

[142] E. Li, H. Wang, and F. Ye, "Two-level multi-surrogate assisted optimization method for high dimensional nonlinear problems," *Applied Soft Computing*, vol. 46, pp. 26–36, 2016.

[143] H. Rabitz and Ö. F. Aliş, "General foundations of high-dimensional model representations," *Journal of Mathematical Chemistry*, vol. 25, no. 2-3, pp. 197–233, 1999.

[144] M. Munetomo and D. E. Goldberg, "Linkage identification by non-monotonicity detection for overlapping functions," *Evolutionary Computation*, vol. 7, no. 4, pp. 377–398, 1999.

[145] W. Chen, T. Weise, Z. Yang, and K. Tang, "Large-scale global optimization using cooperative coevolution with variable interaction learning," in *Parallel Problem Solving from Nature*. Springer, 2010, pp. 300–309.

[146] K. Weicker and N. Weicker, "On the improvement of coevolutionary optimizers by learning variable interdependencies," in *IEEE Congress on Evolutionary Computation*, vol. 3. IEEE, 1999, pp. 1627–1632.

[147] L. Sun, S. Yoshida, X. Cheng, and Y. Liang, "A cooperative particle swarm optimizer with statistical variable interdependence learning," *Information Sciences*, vol. 186, no. 1, pp. 20–39, 2012.

[148] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Nanyang Technological University, Singapore, Tech. Rep., 2005, http://www.ntu.edu.sg/home/EPNSugan.

[149] X. Li, K. Tang, M. N. Omidvar, Z. Yang, and K. Qin, "Benchmark functions for the CEC'2013 special session and competition on large-scale global optimization," RMIT University, Melbourne, Australia, Tech. Rep., 2013, http://goanna.cs.rmit.edu.au/ xiaodong/cec13-lsgo.

[150] H. Ge, L. Sun, X. Yang, S. Yoshida, and Y. Liang, "Cooperative differential evolution with fast variable interdependence learning and cross-cluster mutation," *Applied Soft Computing*, vol. 36, pp. 300–314, 2015.

[151] M. Munetomo and D. E. Goldberg, "Identifying linkage by nonlinearity check," Tech. Rep., 1998.

[152] M. Tezuka, M. Munetomo, and K. Akama, "Linkage identification by nonlinearity check for real-coded genetic algorithms," in *Genetic and Evolutionary Computation Conference*. Springer, 2004, pp. 222–233.

[153] M. Munetomo and D. E. Goldberg, "A genetic algorithm using linkage identification by nonlinearity check," in *IEEE Conference on Systems, Man, and Cybernetics*, vol. 1. IEEE, 1999, pp. 595–600.

[154] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 378–393, 2014.

[155] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM Transactions on Mathematical Software*, vol. 42, no. 2, p. 13, 2016.

[156] Y. Ling, H. Li, and B. Cao, "Cooperative co-evolution with graph-based differential grouping for large scale global optimization," in *International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*. IEEE, 2016, pp. 95–102.

[157] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao, "Dg2: A faster and more accurate differential grouping for large-scale black-box optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 6, pp. 929–942, 2017.

[158] Y. Sun, M. Kirley, and S. K. Halgamuge, "Extended differential grouping for large scale global optimization with direct and indirect variable interactions," in *Genetic and Evolutionary Computation Conference*. ACM, 2015, pp. 313–320.

[159] X.-M. Hu, F.-L. He, W.-N. Chen, and J. Zhang, "Cooperation coevolution with fast interdependency identification for large scale optimiza-

tion," *Information Sciences*, vol. 381, pp. 142–160, 2017.
[160] Y. Sun, M. Kirley, and S. K. Halgamuge, "A recursive decomposition method for large scale continuous optimization," *IEEE Transactions on Evolutionary Computation*, 2017.
[161] M. Yang, A. Zhou, C. Li, and X. Yao, "An Efficient Recursive Differential Grouping for Large-Scale Continuous Problems," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 1, pp. 159–171, Feb. 2021.
[162] K. S. Kim and Y. S. Choi, "An efficient variable interdependency-identification and decomposition by minimizing redundant computations for large-scale global optimization," *Information Sciences*, vol. 513, pp. 289–323, Mar. 2020.
[163] X. Xue, K. Zhang, R. Li, L. Zhang, C. Yao, J. Wang, and J. Yao, "A topology-based single-pool decomposition framework for large-scale global optimization," *Applied Soft Computing*, vol. 92, p. 106295, Jul. 2020.
[164] H.-B. Xu, F. Li, and H. Shen, "A Three-Level Recursive Differential Grouping Method for Large-Scale Continuous Optimization," *IEEE Access*, vol. 8, pp. 141 946–141 957, 2020.
[165] B. Cao, J. Zhao, Y. Gu, Y. Ling, and X. Ma, "Applying graph-based differential grouping for multiobjective large-scale optimization," *Swarm and Evolutionary Computation*, vol. 53, p. 100626, Mar. 2020.
[166] M. N. Omidvar, X. Li, and K. Tang, "Designing benchmark problems for large-scale continuous optimization," *Information Sciences*, vol. 316, pp. 419–436, 2015.
[167] A. Chen, Y. Zhang, Z. Ren, Y. Yang, Y. Liang, and B. Pang, "A global information based adaptive threshold for grouping large scale optimization problems," in *Genetic and Evolutionary Computation Conference*. ACM, 2018, pp. 833–840.
[168] Y. Sun, M. N. Omidvar, M. Kirley, and X. Li, "Adaptive threshold parameter estimation with recursive differential grouping for problem decomposition," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2018, pp. 889–896.
[169] Y. Sun, X. Li, A. Ernst, and M. N. Omidvar, "Decomposition for large-scale optimization problems with overlapping components," in *IEEE congress on evolutionary computation*, 2019.
[170] X. Peng, Y. Jin, and H. Wang, "Multimodal Optimization Enhanced Cooperative Coevolution for Large-Scale Optimization," *IEEE Transactions on Cybernetics*, vol. 49, no. 9, pp. 3507–3520, Sep. 2019.
[171] Z. Ren, A. Chen, M. Wang, Y. Yang, Y. Liang, and K. Shang, "Bi-Hierarchical Cooperative Coevolution for Large Scale Global Optimization," *IEEE Access*, vol. 8, pp. 41 913–41 928, 2020.
[172] J. Fan, J. Wang, and M. Han, "Cooperative coevolution for large-scale optimization based on kernel fuzzy clustering and variable trust region methods," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 4, pp. 829–839, 2014.
[173] L. Li, W. Fang, Q. Wang, and J. Sun, "Differential Grouping with Spectral Clustering for Large Scale Global Optimization," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, Jun. 2019, pp. 334–341.
[174] L. Li, W. Fang, Y. Mei, and Q. Wang, "Cooperative coevolution for large-scale global optimization based on fuzzy decomposition," *Soft Computing*, vol. 25, no. 5, pp. 3593–3608, Mar. 2021.
[175] Z. Yang, K. Tang, and X. Yao, "Multilevel cooperative coevolution for large scale optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2008, pp. 1663–1670.
[176] M. N. Omidvar, Y. Mei, and X. Li, "Effective decomposition of large-scale separable continuous functions for cooperative co-evolutionary algorithms," in *IEEE Congress on Evolutionary Computation*. IEEE, 2014, pp. 1305–1312.
[177] A. Vakhnin and E. Sopov, "Investigation of Improved Cooperative Coevolution for Large-Scale Global Optimization Problems," *Algorithms*, vol. 14, no. 5, p. 146, May 2021.
[178] A. Song, W.-N. Chen, P.-T. Luo, Y.-J. Gong, and J. Zhang, "Overlapped cooperative co-evolution for large scale optimization," in *IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2017, pp. 3689–3694.
[179] W. Fang, L. Zhang, J. Zhou, X. Wu, and J. Sun, "A novel quantum-behaved particle swarm optimization with random selection for large scale optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2017, pp. 2746–2751.
[180] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.
[181] A. Sinha and D. E. Goldberg, "A survey of hybrid genetic and evolutionary algorithms," *IlliGAL report*, vol. 2003004, 2003.
[182] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Ensemble strategies for population-based optimization algorithms – A survey," *Swarm and Evolutionary Computation*, vol. 44, pp. 695–711, Feb. 2019.
[183] L.-Y. Tseng and C. Chen, "Multiple trajectory search for large scale global optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2008, pp. 3052–3059.
[184] V. Gardeux, R. Chelouah, P. Siarry, and F. Glover, "EM323: a line search based algorithm for solving high-dimensional continuous non-linear optimization problems," *Soft Computing*, vol. 15, no. 11, pp. 2275–2285, 2011.
[185] P. Moscato *et al.*, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," *Caltech concurrent computation program, C3P Report*, vol. 826, p. 1989, 1989.
[186] C. Seren, "A hybrid jumping particle swarm optimization method for high dimensional unconstrained discrete problems," in *IEEE Congress on Evolutionary Computation*. IEEE, 2011, pp. 1649–1656.
[187] N. Bouhmala, "A multilevel memetic algorithm for large sat-encoded problems," *Evolutionary Computation*, vol. 20, no. 4, pp. 641–664, 2012.
[188] J.-q. Li and Q.-k. Pan, "Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm," *Information Sciences*, vol. 316, pp. 487–502, 2015.
[189] Y. Mei, X. Li, and X. Yao, "Cooperative coevolution with route distance grouping for large-scale capacitated arc routing problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 435–449, 2014.
[190] R. Shang, K. Dai, L. Jiao, and R. Stolkin, "Improved memetic algorithm based on route distance grouping for multiobjective large scale capacitated arc routing problems," *IEEE Transactions on Cybernetics*, vol. 46, no. 4, pp. 1000–1013, 2016.
[191] N. R. Sabar, J. Abawajy, and J. Yearwood, "Heterogeneous cooperative co-evolution memetic differential evolution algorithm for big data optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 2, pp. 315–327, 2017.
[192] D. Tang, Y. Cai, J. Zhao, and Y. Xue, "A quantum-behaved particle swarm optimization with memetic algorithm and memory for continuous non-linear large scale problems," *Information Sciences*, vol. 289, pp. 162–189, 2014.
[193] S.-Z. Zhao, J. J. Liang, P. N. Suganthan, and M. F. Tasgetiren, "Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2008, pp. 3845–3852.
[194] M. Olguin-Carbajal, E. Alba, and J. Arellano-Verdejo, "Micro-differential evolution with local search for high dimensional problems," in *IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 48–54.
[195] S. Salcedo-Sanz, C. Camacho-Gómez, D. Molina, and F. Herrera, "A coral reefs optimization algorithm with substrate layers and local search for large scale global optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2016, pp. 3574–3581.
[196] M. A. M. de Oca, D. Aydın, and T. Stützle, "An incremental particle swarm for large-scale continuous optimization problems: an example of tuning-in-the-loop (re)design of optimization algorithms," *Soft Computing*, vol. 15, no. 11, pp. 2233–2255, 2011.
[197] C. Liu and B. Li, "Memetic algorithm with adaptive local search depth for large scale global optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2014, pp. 82–88.
[198] A. Bolufé-Röhler, S. Fiol-González, and S. Chen, "A minimum population search hybrid for large scale global optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2015, pp. 1958–1965.
[199] D. Molina, M. Lozano, and F. Herrera, "MA-SW-Chains: Memetic algorithm based on local search chains for large scale continuous global optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 3153–3160.
[200] D. Molina, M. Lozano, A. M. Sánchez, and F. Herrera, "Memetic algorithms based on local search chains for large scale continuous optimisation problems: Ma-ssw-chains," *Soft Computing*, vol. 15, no. 11, pp. 2201–2220, 2011.
[201] I. Fister, I. J. Fister, J. Brest, and V. Žumer, "Memetic artificial bee colony algorithm for large-scale global optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–8.
[202] D. Molina and F. Herrera, "Iterative hybridization of de with local search for the CEC'2015 special session on large scale global optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2015, pp. 1974–1978.
[203] S.-Z. Zhao, P. N. Suganthan, and S. Das, "Self-adaptive differential evolution with multi-trajectory search for large-scale optimization," *Soft Computing*, vol. 15, no. 11, pp. 2175–2185, 2011.

[204] A. Pétrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proceedings of IEEE international conference on evolutionary computation*. IEEE, 1996, pp. 798–803.

[205] H. Rosenbrock, "An automatic method for finding the greatest or least value of a function," *The Computer Journal*, vol. 3, no. 3, pp. 175–184, 1960.

[206] M. J. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The computer journal*, vol. 7, no. 2, pp. 155–162, 1964.

[207] Z. Cao, L. Wang, Y. Shi, X. Hei, X. Rong, Q. Jiang, and H. Li, "An effective cooperative coevolution framework integrating global and local search for large scale optimization problems," in *IEEE Congress on Evolutionary Computation*. IEEE, 2015, pp. 1986–1993.

[208] A. LaTorre, S. Muelas, and J.-M. Peña, "A mos-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test," *Soft Computing*, vol. 15, no. 11, pp. 2187–2199, 2011.

[209] ——, "Multiple offspring sampling in large scale global optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–8.

[210] ——, "Large scale global optimization: Experimental results with MOS-based hybrid algorithms," in *IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 2742–2749.

[211] D. Molina, M. Lozano, and F. Herrera, "Memetic algorithm with local search chaining for large scale continuous optimization problems," in *IEEE Congress on Evolutionary Computation*. IEEE, 2009, pp. 830–837.

[212] E. Segredo, E. Lalla-Ruiz, E. Hart, and S. Voß, "On the performance of the hybridisation between migrating birds optimisation variants and differential evolution for large scale continuous problems," *Expert Systems with Applications*, vol. 102, pp. 126–142, 2018.

[213] Y. Sun, M. Kirley, and S. K. Halgamuge, "A memetic cooperative co-evolution model for large scale continuous optimization," in *Australasian Conference on Artificial Life and Computational Intelligence*. Springer, 2017, pp. 291–300.

[214] J. Tang, M. H. Lim, and Y. S. Ong, "Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems," *Soft Computing*, vol. 11, no. 9, pp. 873–888, 2007.

[215] L. Vitorino, S. Ribeiro, and C. J. Bastos-Filho, "A hybrid swarm intelligence optimizer based on particles and artificial bees for high-dimensional search spaces," in *IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–6.

[216] Y. Wang, B. Li, and T. Weise, "Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems," *Information Sciences*, vol. 180, no. 12, pp. 2405–2420, 2010.

[217] S. Yang and Y. Sato, "Modified bare bones particle swarm optimization with differential evolution for large scale problem," in *IEEE Congress on Evolutionary Computation*. IEEE, 2016, pp. 2760–2767.

[218] S. Ye, G. Dai, L. Peng, and M. Wang, "A hybrid adaptive coevolutionary differential evolution algorithm for large-scale optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2014, pp. 1277–1284.

[219] W. Deng, R. Chen, B. He, Y. Liu, L. Yin, and J. Guo, "A novel two-stage hybrid swarm intelligence optimization algorithm and application," *Soft Computing*, vol. 16, no. 10, pp. 1707–1722, 2012.

[220] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[221] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. of IEEE International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[222] F. J. Solis and R. J.-B. Wets, "Minimization by random search techniques," *Mathematics of operations research*, vol. 6, no. 1, pp. 19–30, 1981.

[223] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[224] A. LaTorre, J. M. Peña, V. Robles, and S. Muelas, "Using multiple offspring sampling to guide genetic algorithms to solve permutation problems," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*. ACM, 2008, pp. 1119–1120.

[225] D. Molina, M. Lozano, C. García-Martínez, and F. Herrera, "Memetic algorithms for continuous optimisation based on local search chains," *Evolutionary computation*, vol. 18, no. 1, pp. 27–63, 2010.

[226] D. Molina, A. LaTorre, and F. Herrera, "SHADE with Iterative Local Search for Large-Scale Global Optimization," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, Jul. 2018, pp. 1–8.

[227] A. LaTorre, S. Muelas, and J.-M. Peña, "Evaluating the multiple offspring sampling framework on complex continuous optimization functions," *Memetic Computing*, vol. 5, no. 4, pp. 295–309, 2013.

[228] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. IEEE, 2008, pp. 1110–1116.

[229] A. A. Hadi, A. W. Mohamed, and K. M. Jambi, "Lshade-spa memetic framework for solving large-scale optimization problems," *Complex & Intelligent Systems*, vol. 5, no. 1, pp. 25–40, 2019.