A Review of Population-Based Metaheuristics for Large-Scale Black-Box Global Optimization—Part II

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

***Abstract*—This article is the second part of a two-part survey series on large-scale global optimization. The first part covered two major algorithmic approaches to large-scale optimization, namely, decomposition methods and hybridization methods, such as memetic algorithms and local search. In this part, we focus on sampling and variation operators, approximation and surrogate modeling, initialization methods, and parallelization. We also cover a range of problem areas in relation to large-scale global optimization, such as multiobjective optimization, constraint handling, overlapping components, the component imbalance issue and benchmarks, and applications. The article also includes a discussion on pitfalls and challenges of the current research and identifies several potential areas of future research.**

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

I. Introduction

The FIRST part of this two-part survey series covered decomposition methods and hybrid methods as two most widely investigated approaches in the literature. Fig. 1 depicts a high-level structure of the main topics covered across both parts. In this part, we review more approaches to large-scale global optimization and also address several problem areas, including multiobjective optimization and constraint handling. Section II covers the sampling mechanism and variation operators of two well-known algorithms: 1) particle swarm algorithm [1] and 2) differential evolution [2], and how they are modified to solve large-scale problems. Section III covers the algorithms that rely on some form of approximation to cope with the challenges of high dimensionality. Section IV covers population initialization methods and their significance in the large-scale global optimization. Section V addresses the role of parallel algorithms to address the issue of scalability.

In addition to the algorithmic approaches to large-scale optimization, the article also addresses a range of problem areas pertaining to large-scale global optimization. These include: 1) scalability of multiobjective optimization algorithms with respect to their decision space; 2) challenges of constraint handling in the context of large-scale optimization; 3) challenges in dealing with problems with overlapping components and the issue of exploitable structure; 4) resource allocation and the problem of imbalanced contribution; and 5) benchmarking and application areas.

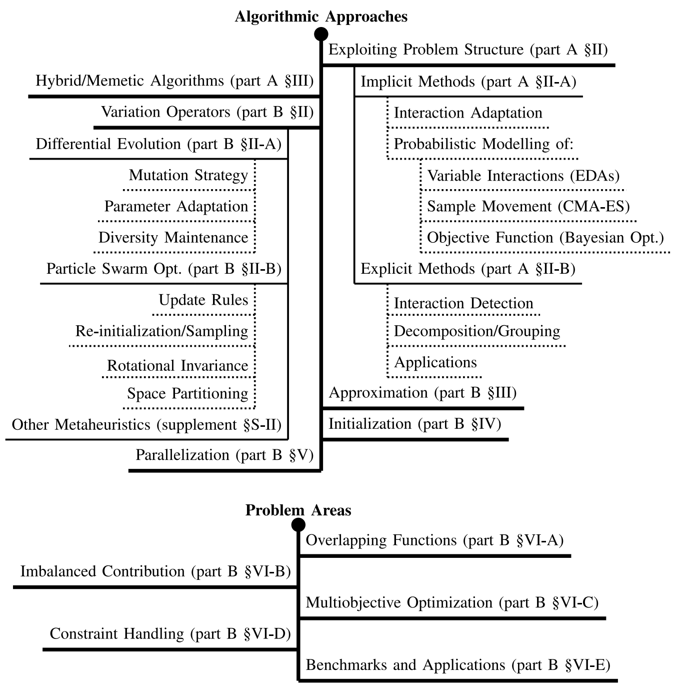
Algorithmic Approaches
Exploiting Problem Structure (part A §II)
Hybrid/Memetic Algorithms (part A §III)
Implicit Methods (part A §II-A)
Variation Operators (part B §II)
Interaction Adaptation
Differential Evolution (part B §II-A)
Probabilistic Modelling of:
Mutation Strategy
Variable Interactions (EDAs)
Parameter Adaptation
Sample Movement (CMA-ES)
Diversity Maintenance
Objective Function (Bayesian Opt.)
Particle Swarm Opt. (part B §II-B)
Explicit Methods (part A §II-B)
Update Rules
Interaction Detection
Re-initialization/Sampling
Decomposition/Grouping
Rotational Invariance
Applications
Space Partitioning
Approximation (part B §III)
Other Metaheuristics (supplement §S-II)
Initialization (part B §IV)
Parallelization (part B §V)
Problem Areas
Overlapping Functions (part B §VI-A)
Imbalanced Contribution (part B §VI-B)
Multiobjective Optimization (part B §VI-C)
Constraint Handling (part B §VI-D)
Benchmarks and Applications (part B §VI-E)

Fig. 1. Outline of the topics covered in the two parts of this survey series on large-scale global optimization.

Manuscript received 26 March 2021; revised 17 September 2021; accepted 9 November 2021. Date of publication 25 November 2021; date of current version 3 October 2022. This work was supported in part by the Australian Research Council (ARC) Discovery under Grant DP180101170 and Grant DP190101271; in part by the Shenzhen Science and Technology Program under Grant KQTD2016112514355531; in part by the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant 2017ZT07X386; and in part by the Program for University Key Laboratory of Guangdong Province under Grant 2017KSYS008. *(Corresponding author: Mohammad Nabi Omidvar.)*

Mohammad Nabi Omidvar is with the School of Computing and Leeds University Business School, University of Leeds, Leeds LS2 9JT, U.K. (e-mail: m.n.omidvar@leeds.ac.uk).

Xiaodong Li is with the School of Computing Technologies, RMIT University, Melbourne, VIC 3000, Australia (e-mail: 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, and also with the School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K. (e-mail: xiny@sustech.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TEVC.2021.3130835.

Digital Object Identifier 10.1109/TEVC.2021.3130835

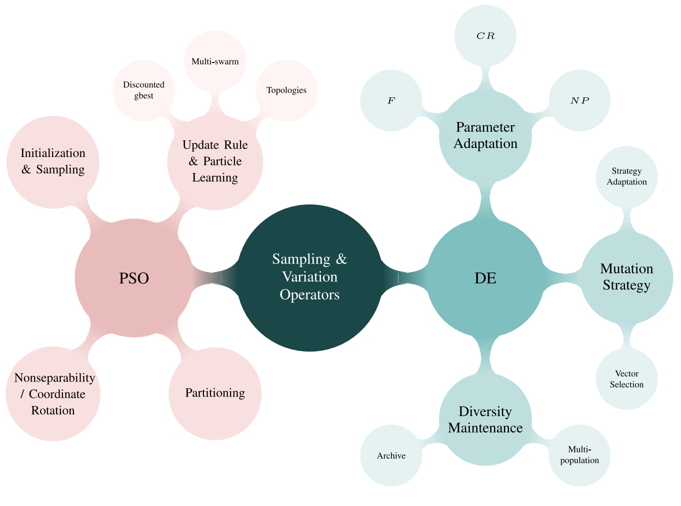

Fig. 2. DE and PSO are two widely used metaheuristic algorithms used in large-scale global optimization. This figure shows major aspects of these algorithms studied for large-scale optimization problems.

The article also features a section on pitfalls and challenges of the field and potential areas of future research.

II. Sampling and Variation Operators

In part I of this survey, we have seen that many optimizers, such as estimation of distribution algorithms (EDAs), differential evolution (DE), and particle swarm optimization (PSO), can be used as component optimizers in decomposition-based frameworks (part I Section II-B), and as explorative agents in memetic algorithms (part I Section III). In this section, we focus on algorithm-specific aspects, such as parameter adaptation, modification of variation operators or design of new ones, diversity maintenance mechanisms, etc. In what follows, we cover DE and PSO in more detail and other metaheuristics are covered in Section S-II of the supplementary material. EDAs were covered in Part I Section II-A due to their focus on modeling variable interactions. Fig. 2 shows major aspects of PSO and DE, which have been studied under high-dimensional settings.

A. Differential Evolution

Due to its versatility, ease of implementation, and simplicity, DE [2] has become a widely used optimization algorithm for global optimization [3]. Consequently, many variants of DE have been developed for large-scale global optimization [4] from which the most popular ones are briefly reviewed in this section. Most of the DE variants proposed for large-scale optimization are centered around maintaining population diversity, which is done by various means, such as parameter adaptation, modification of DE mutation strategy, and diversity maintenance mechanisms.

1) Mutation Strategy: Mutation strategy is DE's central variation operator and has been subject to extensive investigation in [3]. Several attempts have been made to improve DE for large-scale optimization by adapting or hybridizing several mutation strategies or by proposing new ones [5].

a) Adaptation of mutation strategy: Different mutation strategies exhibit various degrees of explorative/exploitative power each being suitable for certain problem types [5]. In the context of LSGO, several attempts have been made to use several mutation strategies to improve the convergence properties of DE in high-dimensional spaces. These methods are either based on adaptively applying a set of strategies to a single population or using a multipopulation approach where each is evolved using its own mutation strategy. Ali *et al.* [6] proposed a multipopulation DE, where each subpopulation has its own mutation strategy. Banitalebi *et al.* [7] proposed a binary DE, which adaptively selects the mutation strategy for generating trial vectors and also adapts the scaling factor and crossover rate using a chaotic process (also see Section II-A2). Kushida *et al.* [8] proposed a rank-based mechanism for selecting the mutation strategies. Wang *et al.* [9] proposed to adaptively switch between DE/rand/1 and DE/current_to_best/1 mutation strategies. There are also approaches that switch between DE/rand/1/bin and a newly proposed strategy based on a uniform distribution [10], [11].

b) Vector selection: Canonical DE uses random individuals in the mutation process to generate a scaled difference vector to be applied to a base vector and generate a new solution. The choice of the vectors participating in the mutation procedure plays a crucial role in DE's convergence behavior [5]. Ge *et al.* [12] analyzed different DE strategies and observed that those using the best individual are exploitative while those using random individuals are more explorative. They argue that instead of randomly selecting the participating vectors, it is better to systematically choose a vector close to the best solution to favor exploitation and far from the mutant to favor exploration. Inspired by PSO personal and global-best particles, Wang *et al.* [13] proposed to generate trial vectors by including the global-best and personal-best individuals in the mutation strategy. The authors claim that this process is akin to neighborhood search and improves convergence. García-Martínez *et al.* [14] associated four basic roles—1) placing; 2) leading; 3) correcting; and 4) receiving—to each vector (solution), and the vector selection for mutation is performed based on these four basic roles. The vector selection strategy proposed by Ali *et al.* [6] is a function of the rank of a solution in the population, favoring higher quality solutions to participate in the mutation. Zhang and Sanderson [15] proposed a generalization of the classic DE/current-to-best mutation operator, DE/current-to-*p*best, which uses the top *p*% best solutions to balance the greediness level of the algorithm and to maintain better diversity in the population. Some other studies also proposed several mutation strategies in which the solution quality is taken into account in the vector selection process [10], [11]. Yang *et al.* [16] proposed to use multiple such difference vectors that are scaled differently to generate trial vectors.

The choice of the base vector to which the mutation is applied is also important in DE's convergence behavior. Ali *et al.* [6] proposed a new mutation strategy by selecting the base individual to be a convex combination of randomly chosen individuals from the population. Wang *et al.* [17] proposed an enhanced opposition-based DE in which the candidate solutions are translated into a so-called *opposite space* using the definition of opposite numbers [18]. Wang *et al.* [17]

argued that by evaluation of the candidate solutions and their translated counterparts in the opposite space, the probability of finding better solutions increases. This hypothesis is backed up by a set of empirical results on a set of 19 high-dimensional benchmark functions [19]. Hiba *et al.* [20] proposed a center-based mutation strategy that uses the center of three randomly chosen solutions as the base vector.

2) Parameter Adaptation: Population size (*NP*), crossover rate (*CR*), and the scaling factor (*F*) are DE's major parameters affecting its convergence properties on different problem types. To eliminate the need for practitioners to set these hard-to-tweak parameters, several attempts have been made to adaptively set these parameters in the course of optimization [3]. In this section, we review some of these adaptation methods pertaining to large-scale global optimization.

The scaling factor and crossover rate are the two most studied parameters. Most of these attempts use some form of probability distribution from which the parameters are sampled. Brest *et al.* [21] dynamically changed *F* and *CR* using a uniform distribution. Wang *et al.* [22] used a similar adaptation mechanism except that they restricted the range of *CR* values. Brest *et al.* [23] introduced a sign changing mechanism to *F* in addition to sampling of *CR* and *F* values from a uniform distribution. To improve the current best, it uses smaller *F* values in the second half of the optimization process. Weber *et al.* [24] used a multipopulation approach each having its own scaling factor, which is regenerated in a probabilistic way. Zamuda *et al.* [25] proposed to adapt *CR* and *F* using a log-normal distribution. Improving upon self-adaptive DE (SaDE) [26], Yang *et al.* [27] used a Gaussian distribution to generate *F* and *CR* for each individual and update the mean of the Gaussian based on the parameter values succeeding in generating surviving offsprings. Zhang and Sanderson [15] proposed JADE, which randomly generates *F* and *CR* at every generation using Cauchy and Gaussian distributions, respectively, whose parameters are adapted in the course of optimization. Yang *et al.* [28] attempted to generalize the attempts by its predecessors, such as JADE, SaDE, and SaNSDE into a unified mechanism.

There are also alternative approaches that are not based on sampling from probability distributions. For example, some studies propose to change the crossover rate and the scaling factor using a chaotic process [7], [9]. Takahama and Sakai [29] proposed a DE variant in which the scaling factor is adapted according to the modality feature of the search space. Kushida *et al.* [8] improved upon the works of Takahama and Sakai [29] by adding a rank-based mechanism for setting the scaling factor and crossover rate, as well as the mutation strategies.

The attempts for adaptation of population size in large-scale optimization are *ad hoc* and limited. Brest *et al.* [23] proposed to gradually reduce the population size in the course of optimization. Wang *et al.* [30] adaptively changed the population size by adding or removing solutions based on their performance. Tanabe and Fukunaga [31] linearly decreased the population size.

3) Diversity Maintenance: Loss of population diversity is central to DE's deficiency in high-dimensional spaces. This is typically avoided in lower dimensions by means of increasing the population size [32]. However, in high-dimensional spaces, the large population size hinders convergence [33]. The adaptation of the mutation scaling factor, hybridizing an array of mutation strategies, or designing new ones are all attempts to improve the population diversity, which were addressed in the previous sections. Other approaches to diversification are *multipopulation* approaches, either in the form of several islands searching the original search space or by means of partitioning and coevolution, or maintaining an *archive* of solutions (Fig. 2).

Weber *et al.* [24] proposed a multipopulation strategy that simultaneously searches different parts of the search space. This algorithm randomly rearranges the individuals across the subpopulations with the aim of maintaining diversity among the solutions. Ge *et al.* [34] also proposed a multipopulation DE, which maintains diversity through migration of similar or diverse individuals. This mechanism controls the balance between exploration and exploitation. Ge *et al.* [35] used a multipopulation approach with automatic merge and split operations to improve the population diversity. Ali *et al.* [6] used a multipopulation DE with each population having its own mutation strategy to maintain population diversity. The information exchange between populations helps with balancing exploration and exploitation. Parsopoulos [36] used cooperative coevolution to partition the search space into smaller regions and optimized them with a micro DE. Micro DE is prone to losing diversity and getting trapped in the local optima; however, CC helps DE to focus the search with its micro population on smaller regions. Ge *et al.* [12] also used CC with cross-cluster mutation to promote exploration.

Maintaining an archive of solutions in the course of optimization is another means of maintaining diversity. Takahama and Sakai [29] proposed a DE variant with an archive of old solutions to help with diversification in the mating process, especially when the population size is small. Yang *et al.* [16] also proposed a DE variant, which keeps an archive of failed trial vectors with the hope of preserving the good genetic material. Zhang and Sanderson [15] proposed JADE, which maintains an external archive of inferior solutions to estimate the possible improvement directions. The external archive of JADE proved to be beneficial, especially on relatively high-dimensional problems with up to 100 dimensions.

B. Particle Swarm Optimization

PSO [1], [37] is known to be susceptible to premature convergence, which is magnified on high-dimensional problems [38], [39]. Most approaches to handle large-scale optimization problems are centered around increasing diversity to improve exploration. In some cases, however, extreme exploration and exploitation can co-exist [40]. The common remedies to PSO's premature convergence in the large-scale global optimization literature are population reinitialization, complementary sampling mechanisms, population size adaptation, space partitioning (by means of cooperative coevolution or otherwise), improving PSO's update rule and particle learning mechanisms, and mechanism to deal with variable interaction and nonseparable problems.

1) PSO Update Rule and Particle Learning: Excessive reliance on the global-best particle can result in premature convergence. Many attempts to avoid premature convergence revolve around reducing the influence of global best. Cheng and Jin [41] proposed a PSO variant, named CSO, which does not use personal or global-best solutions to update the position of the particles. Instead, random pairs are chosen to compete and the winner returns directly to the swarm, while the loser is updated by learning from the winner. CSO maintains a better diversity than PSO and is more explorative, making it better suited for large-scale global optimization. Tian *et al.* [42] proposed a variant of CSO based on a two-stage update rule for the position of particles and applied it to solving multiobjective problems. Naderi *et al.* [43] proposed a fuzzy adaptive system to adjust the inertia weight. This eliminates the use of global best in the velocity update rule to avoid premature convergence. Tang *et al.* [44] used a new update rule to exploit four best positions via Gaussian sampling to reduce the influence of global best and promote exploration. Pluhacek *et al.* [45] changed the velocity update rule such that with some probability, the velocity is either zero or is updated by taking either a random particle, personal best, or the global best into account.

Controlling information exchange among particles by means of population topologies or multipopulation structures are other ways of enhancing the swarm diversity [46]. Fan *et al.* [47] proposed a PSO variant, which builds a dynamic neighborhood topology for PSO by performing clustering on the population. The neighbors of the particles are chosen from the same cluster. It also chooses a distant neighbor for particles through a random selection. Zhang *et al.* [48] improved CSO by applying Cauchy and Gaussian updates on the winner particles and used a ring topology to enhance the swarm diversity. Distributed multipopulation schemes [49]–[51] also promote controlled information exchange among particles, which can improve the population diversity.

In addition to the above, several other modifications to PSO's update rule have been suggested in the context of large-scale global optimization. Arasomwan and Adewumi [52] found that inertia weight, acceleration coefficients, and random factors were not of significance in velocity update for obtaining global solutions. They proposed to adaptively update particles' velocity based on the Euclidean distance between particles and the global best. It also introduces the chaotic behavior into the particle position update rule. The notion of social learning has been proposed to reduce the adverse effect of isolated asocial learning [53], [54]. Yang *et al.* [55] proposed to group particles into several levels based on their fitness. Two predominant particles from two different higher levels are chosen to guide the learning of particles. This has shown to improve diversity. The convergence speed controller was proposed as an independent operator to respond to premature or slow convergence [56]. Cheng *et al.* [57] proposed a mutation operator based on the alpha-stable distribution to enhance the swarm diversity and avoid premature convergence. Li *et al.* [58] changed the particles' velocity update rule to decouple exploration and exploitation. Xue *et al.* [59] used multiple velocity and position update rules that are chosen probabilistically, whose parameters are adapted according to the effectiveness of each strategy.

2) Reinitialization, Sampling, and Population Size Control: Hsieh *et al.* [60] proposed a PSO variant with dynamic swarm size, which increases or decreases the swarm size based on the status of particles. In general, if the global best of the swarm is not updated for several consecutive iterations, new particles are generated by applying a crossover-like operator on the best solutions that were obtained in the past. Conversely, if the information content of the swarm is rich enough to allow frequent and robust updating of the global best, some of the poor-quality solutions might be removed from the swarm. There are some other mechanisms in place to avoid the growth of the swarm size beyond bounds. De Oca *et al.* [61] proposed an improved version of incremental particle swarm-guided local search [62], which incrementally increases the population size to solve large-scale continuous optimization problems.

Garcí-Nieto and Alba [63] proposed restart PSO with velocity modulation. Velocity modulation is the process by which the particles are guided within a region of interest. Additionally, a restart mechanism is devised to avoid premature convergence of the algorithm. Cheng *et al.* [64] proposed partial reinitialization to improve exploration. They partition the search space and count the number of particles in each partition and abandon the low-activity areas. It also subdivides and reinitializes particles in higher activity regions. This mechanism has shown to improve exploitation. Zhou *et al.* [65] introduced opposition-based sampling into CSO.

3) Nonseparability and Coordinate Rotation: The update equations of PSO are dimensionwise, which makes it suitable for separable functions. Hendtlass [66] used dynamic momentum values to enable PSO to better handle nonseparability, making it suitable for functions with interacting dimensions. Korenaga *et al.* [67] introduced the coordinate rotation into the velocity update rule, which makes it possible to consider the information of other coordinates when calculating the velocity of a component. This can improve population diversity and has shown to be beneficial for large-scale optimization. Chu *et al.* [68] used PCA to parts of the space not spanned by the current population resulting in improved exploration. In a similar way, Chu *et al.* [69] also used PCA to find lost dimensions and promote search in the less explored areas due to lost dimensions.

4) Space Partitioning: Zhang *et al.* [70] partitioned the space by grouping the dimensions into segments. Then, some newly designed operators are assigned to each segment to update those segments. These operators are designed to improve population diversity and avoid premature convergence. Zhao *et al.* [71] proposed to form subswarms which work independently during the search. Then, the subswarms are randomly changed to enlarge their neighborhood and promote information exchange and population diversity. This improves exploration but may deteriorate exploitation, which is compensated for by means of local search. Cheng *et al.* [64] proposed space partitioning with the aim of identifying and abandoning low-activity areas and focused the search effort in high-activity areas.

Balancing exploration and exploitation and maintaining population diversity are generally central to the design of effective optimizers for large-scale optimization. In this section, we have seen that this plays a crucial role in both DE

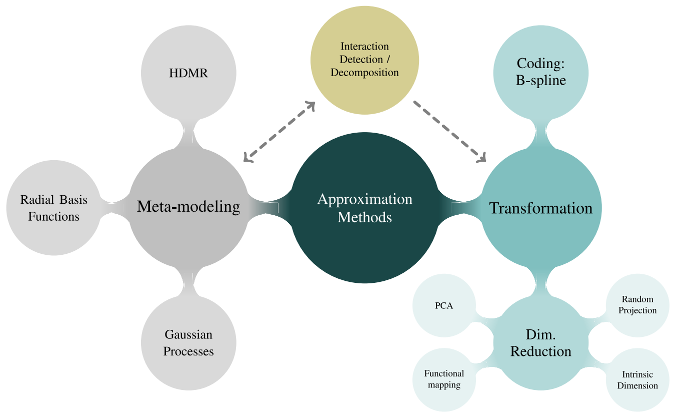

Fig. 3. Problem approximation/simplification methods used for large-scale global optimization.

and PSO. Novel means of using population topologies [46], sampling methods, parameter adaptation, and space partitioning are needed to further improve these algorithms for large-scale optimization.

III. Approximation and Surrogate Modeling

Solving an approximation of a problem can potentially be more viable than obtaining a solution for its original high-fidelity model. In other words, the aim of approximation is to *simplify*. In optimization, this simplification is either achieved by means of applying some *transformation* to the objective function (often to reduce the dimensions), or by building a model of the objective function or its constraints [72], i.e., a *metamodel* to act as a surrogate to the original complex problem (Fig. 3). Metamodels or surrogates are used to reduce the computational overhead of optimizing expensive objective functions. In recent years, one approach to large-scale global optimization is to treat it as an expensive optimization problem and use metamodels to solve it [73].

Metamodels are built and refined based on sampling of the objective function. In high-dimensional spaces, the accuracy of the model drops significantly due to the limited sample size upon which the model is built. To alleviate this problem, several studies use some form of problem decomposition to break the problem into a set of lower dimensional subproblems, each of which is approximated using a metamodeling technique, such as radial basis functions or the Gaussian processes. Due to problem decomposition, it is clear that the problem structure and variable interaction plays an important role in building an ensemble of surrogates. In some cases, the metamodeling itself is used to identify separable and nonseparable components of a problem [74]. For example, [75] used cut-HDMR to detect the components [75]. Then, a multisurrogate strategy is used to model the nonseparable components. In other cases, variable interaction analysis algorithms, such as differential grouping [76] is used to decompose the problem and the subsequent subproblems are then approximated using metamodeling techniques [77]. Werth *et al.* [78] also proposed a sliding window approach over the decision vectors to reduce the dimensionality of the problem. They use LINC-R to discover the variable interaction structure of a subset of the decision variables falling within the sliding window. Then, a surrogate-assisted algorithm is used to optimize over those variables.

Metamodeling and problem decomposition are mutually benefiting approaches to solve large-scale optimization problems. Problem decomposition makes it possible to build more accurate metamodels, given the limited samples, while metamodels can help with the issue of evaluating partial solutions in a divide-and-conquer paradigm. In cooperative coevolution and other divide-and-conquer paradigms, partial solutions need to be evaluated in the context of other partial solutions to form a complete solution. The issue of estimating the fitness of partial solutions was studied by Wang and Gao [79] using fixed auxiliary functions with no dynamic metamodeling mechanism. Several studies use metamodeling as a means of reducing the overhead of matching partial solutions and their re-evaluation for cooperative coevolution [77], [80], [81] and other divide-and-conquer paradigms, such as random projections [73].

In addition to modeling of subproblems, metamodels have also been used to balance the global search (exploration) and local search (exploitation) efforts. Metamodels have been used in the competitive swarm optimizer [41] to approximate the fitness of neighboring particles of a particle with a known fitness [82]. Sun *et al.* [83] proposed to balance the exploration and exploitation efforts by means of building local and global surrogates. For the exploration part, they use social learning PSO [53], which has good global optimization properties in conjunction with radial basis functions capable of capturing the global profile of the objective function. For exploitation, they use a variant of the fitness approximation method proposed by Sun *et al.* [82] in conjunction with PSO for local search.

Beside building an explicit model of the objective function, as is the case with metamodeling, approximation can be built into problem representation [84] or be achieved by means of dimensionality reduction through transformation [85]–[91] or finding intrinsic dimensions of a problem [92]. Wang *et al.* [93] combined the benefit of metamodels and dimensionality reduction by using autoencoders to find lower dimensional features of graph embedding problems and use them to construct a surrogate model to approximate the robustness value of large-scale graph networks. Principal component analysis has also been used to identify a lower dimensional representation of the probabilistic Gaussian models of EDAs [88], and the convergence variables on multiobjective problems [89]. The variable reduction strategy is another means of representing the decision variables of an objective function or its constraints based on a smaller subset of *core* decision variables [90], [91]. The random projection theory, which was covered in part I of the survey as an implicit way of exploiting the problem structure, can be used for dimensionality reduction in the context of EDAs. The weighted optimization framework, inspired from adaptive weighting by Yang *et al.* [94], is another way of transforming the problem into a lower dimensional one (see Section VI-C).

IV. Initialization Methods

The random initialization of a set of candidate solutions is at the heart of the existing metaheuristic algorithms. The aim of initialization methods is to make the best use of random number generators or other sampling techniques to cover the vast search space more uniformly. This section covers the studies on random initialization for large-scale global optimization.

A wide range of population initialization methods have been employed by evolutionary algorithms [95], [96]. There are various conclusions, conflicting at times, on the effect of initialization methods on large-scale optimization [97]–[99]. Kazimipour *et al.* [97] studied the effect of advanced initialization methods on large-scale optimization. The study suggested that EAs are more sensitive to initialization in high-dimensional spaces than in lower dimensional ones, regardless of the population size. They also reported that the pseudorandom number generator is inferior to advanced initialization methods in high dimensions. A follow-up study showed that the effect of advanced initialization methods becomes marginal when the parameters of the optimizer are properly set [98]. A systematic study of advanced initialization methods on a DE variant, DE/rand/1/bin [3], showed that when the parameters of the algorithm are set close to their optimal, the statistical difference between random number generators and other initialization methods becomes insignificant.

Segredo *et al.* [100] showed that although overall distinction between random number generators and advanced initialization methods fades away in high-dimensional spaces, there is still a significant difference between them when best case and worst case performances are taken into account. They therefore concluded that the choice of the initialization method is of crucial importance, especially when a limited number of runs are allowed.

Kazimipour *et al.* [99] used centered L_2 discrepancy to measure population uniformity as a function of population size and the dimensionality of the space. They reported that the loss of population uniformity (hence diversity) due to curse of dimensionality is the dominant factor in the performance degradation of optimization algorithms, regardless of the choice of the initialization method. Putting differently, it is the geometric peculiarities of high-dimensional spaces that affect all initialization methods. For example, it is well known that the contrast in the distance between randomly chosen points diminishes as the dimensionality of the space increases [101], [102], which has serious implications on various initialization/sampling techniques. Consequently, Kazimipour *et al.* [99] recommended the use of advanced initialization methods only when the population size and the problem dimensionality are low.

V. Parallelization

In this section, we review the algorithms that rely on CPU and GPU parallelization to improve solving large-scale global optimization problems.

A. Historical Context

Cantú-Paz and Goldberg [103] studied the scalability of parallel single- and multipopulation GAs. Their goal was to find the optimal number of processors that minimizes the runtime. Their analysis showed that the number of processors that minimizes the execution time is proportional to the square root of the population size and the objective function evaluation time. Munetomo *et al.* [104] also investigated the use of parallel processing for linkage learning and proposed a parallel implementation of the LINC linkage learning algorithm called pLINC (see part I of the survey). They also proposed a two-level GA with a series of intra-GAs operating on the linkage groups identified by pLINC, and an inter-GA that operates at a higher level and treats the linkage groups as a whole.

B. Large-Scale Cases

In the context of large-scale optimization, two types of parallelization are common. The first type is specific to a particular EA, and the second type is a generic framework applicable to a wide range of EAs most of which are based on a divide-and-conquer paradigm by means of problem decomposition.

In the algorithm-specific department, Mendiburu *et al.* [105] proposed a parallel master–slave implementation of several binary and continuous EDAs based on a Bayesian network model using message passing interface (MPI) and POSIX threads. They parallelized the learning phase or the model-building process, which often takes the maximum proportion of the execution time and tested their algorithm on 500-D binary problems and 1500-D continuous problems. A drawback of this study is the use of very simple benchmark problems, such as OneMax for the binary case and the sphere function, which is fully separable, for the continuous case. Wang *et al.* [22] proposed a parallelized version of DE based on GPU parallelization and tackled continuous problems of up to 1000 dimensions. They also observed that with a fixed population size the speed-up rate decreases as the dimensionality of the problem increases. Iturriaga and Nesmachnow [106] proposed a parallel version of compact GA for CPU/GPU architectures and tested it on OneMax and noisy OneMax with up to one billion variables. They also proposed an asynchronous model on GPU, which is only suitable for large-scale separable problems. More recently, Duan *et al.* [107] proposed a spark-based software framework for parallelization of various PSO implementations. Their proposed parallel PSO showed a superlinear speedup and was tested on continuous benchmark functions with up to 10^5 dimensions as well as on expensive functions. Cao *et al.* [108] proposed a parallel quantum-enhanced DE by parallelizing the fitness evaluation of individuals.

Lastra *et al.* [109] proposed a GPU-based MA-SW-Chains [110], a memetic algorithm for large-scale global optimization (see part I of the survey), by parallelizing all major components of the algorithm, such as fitness function evaluation, crossover, local search, random number generation, and population sorting. In another study, Cano and García-Martínez [111] proposed an improved GPU-based model for MA-SW-Chain and tested it on a scaled version of the CEC'2013 large-scale benchmarking suite on functions with up to 100 million decision variables. Cano *et al.* [112]

proposed a parallel MA-SW-Chains by adapting it to the MapReduce framework to tackle problems with up to 10 million decision variables. The local search component of the algorithm is done using a divide-and-conquer strategy by performing local search for a subset of the decision variables.

Multipopulation algorithms are common in parallelizing many metaheuristics for large-scale optimization. Ge *et al.* [35] proposed a multipopulation topology-based island model with a master–slave paradigm to solve large-scale problems. Wang *et al.* [50] proposed a distributed PSO based on randomly formed equally size subpopulations, which are co-evolved using a master–slave paradigm. Yang *et al.* [49] proposed a distributed swarm optimizer-based multipopulation master–slave model where the elites of each subpopulation are used in the velocity update rule. Su *et al.* [113] proposed a parallel multiobjective algorithm for community detection. They first identify the key nodes in the network graph and the communities associated with the key nodes are then detected in parallel using a multipopulation model.

Problem decomposition into lower dimensional subproblems is central in several recent parallelization frameworks. Among them, Cao *et al.* [114] proposed a distributed parallel cooperative coevolutionary algorithm for solving large-scale multiobjective problems. They used a variant of differential grouping [115] to decompose the decision space into smaller components, which are optimized in parallel using a two-level parallelization structure based on the MPI. The experimental results are based on 1000-D DTLZ and WFG test functions. De Falco *et al.* [80] proposed a decomposition-based parallel model for solving expensive large-scale continuous optimization problems. They use the random grouping decomposition method and build a separate surrogate (metamodel) for each component of the problem. The components are then solved in parallel using cooperative coevolution. The proposed algorithm was tested on problems with up to 1000 dimensions. Yang *et al.* [116] argued that decomposition-based methods, despite their modular nature, cannot be readily parallelized due to the defects in how partial solutions are evaluated. They show that the objective function used to assign a fitness to a partial solution is not consistent with the ideal fitness assignment. They address the problem of fitness assignment to partial solutions for divide-and-conquer methods by appealing to a parallel framework called naturally parallelizable divide and conquer.

Decomposition-based and multipopulation parallelization methods can also be combined to solve large-scale problems. Hybrid two-way parallelism has also been used for large-scale optimization. These often combine: 1) parallelism by means of problems decomposition, i.e., the problem is decomposed into several lower dimensional subproblems and 2) parallelism by means of population distributions, i.e., a distributed pool model [117]. Jia *et al.* [118] proposed such a two-way algorithm that controls the resource allocation by adapting the subpopulation sizes as well as the number of iterations a particular component (subproblem) is optimized. Another two-way parallelism decomposes the problem into several components based on the variable interaction analysis. Each component is subsequently divided into subpopulations each receiving a processor for optimization. A resource allocator then prioritizes processor allocation as a function of components' contribution toward the overall improvement of the objective function.

VI. Related Research Topics

In the previous sections, we reviewed common *approaches* to large-scale global optimization. In this section, however, we take a problem oriented perspective and discuss several problem areas arising in the context of large-scale global optimization, such as the problem of overlapping components (Section VI-A), resource allocation and the imbalance problem (Section VI-B), decision space scalability of multiobjective optimization (Section VI-C), and the scalability of constrained optimization problems (Section VI-D). The section concludes with a discussion on benchmarking large-scale optimization algorithms and a brief review of their real-world applications (Section VI-E).

A. Overlapping Problems

The importance of the problem structure and how it can be exploited in various ways were discussed in part I of the survey. The decomposition approaches covered in part I of the series are mostly suited to partially separable problems, i.e., those with distinct independent lower dimensional components. However, there are problems with the sparse variable interaction structure, which are not partially decomposable. These problems, which we refer to as *overlapping* problems, occur in many application areas, such as multidisciplinary design optimization [119] and concurrent engineering [120]. Multiobjective optimization problems can also be seen as overlapping problems due to shared decision variables among the objective functions [121]. This is particularly the case when a scalarization technique is used to convert them to a series of single objective optimization problems. Constrained optimization problems may have overlapping interaction structures [122] or may become overlapping depending on the constraint-handling techniques used to handle them. The variable interaction structure of the overlapping problems can be discovered using the methods outlined in part I of the survey. However, exploiting the structure is a more challenging task as compared to partially separable problems. Despite the importance of overlapping problems, very few works have been dedicated to large-scale overlapping problems [78], [123], [124]. This section reviews some of such techniques that can help with the scalability of algorithms for large-scale global optimization.

Some approaches rely on breaking selected interactions with the aim of converting the overlapping problems into partially separable ones [125]–[127] or by means of special crossover operators that take the overlapping nature of the problem into account [128], [129]. Munetomo and Goldberg [125] used monotonicity checking to identify the variable interaction structure of the objective function and proposed a metric to measure the linkage tightness with the aim of breaking weak interactions. Yu *et al.* [130] used an information-theoretic interaction detection mechanism to detect the problem structure and used an entropy-based measure of a component or building block to identify and break weak interactions. Yu *et al.* [131] assumed that the interaction structure is given and designed a crossover operator that partitions the interaction graph into two subgraphs such that the disruption of overlapping components is minimized. Sun *et al.* [126] proposed a variation of recursive differential grouping [132], RDG3, which limits the dimensionality of components forcing some interactions to be broken as a consequence. Li *et al.* [127] proposed a decomposition method based on spectral clustering, which takes the strength of interactions into account and breaks some weak interactions such that intergroup interactions are minimized and the intragroup interactions are maximized.

Thierens [128] proposed to use hierarchical clustering to represent the interactions using a linkage tree, which is subsequently used to perform recombination. Bosman and Thierens [129] proposed an improved version of linkage tree recombination to eliminate superfluous hierarchical linkage relations. Experimental results have shown that this type of recombination performs well on overlapping problems.

In the context of cooperative coevolution, overlapping components or groups with shared decision variables is a common way of solving overlapping problems. Sun *et al.* [123] used monotonicity detection to identify the structure of an n-dimensional problem and form n groups, one for each decision variable containing all other variables it interacts with. These n groups, which are not necessarily mutually exclusive, are optimized in a round-robin fashion within a CC framework. Due to the nonexclusive nature of the groups, this algorithm takes, to some extent, the overlapping nature of the problem into account. Jia *et al.* [133] used the variable interaction analysis to identify the underlying components of the objective function and their shared variables. They then use a contribution-based mechanism to promote components with a higher contribution toward improving the objective function and assign the shared variables the components with large contributions.

Werth *et al.* [78] used a sliding window mechanism and optimized the variables inside the window to reduce the dimension of the problem. The problem structure is taken into account by iteratively constructing the interaction matrix of the problem using LINC-R; however, instead of finding the entire matrix, only the interactions within a given sliding window are considered at each iteration. To deal with overlap, the Cuthill–McKee algorithm is used to reduce the bandwidth of the matrix, which places interacting variables close to each other. Song *et al.* [124] proposed overlapped cooperative coevolution that uses a mechanism called delta disturbance to find the most influential variables and distribute them among the existing components. Although this algorithm was not intended for overlapping problems, the replication of influential variables within all components can have a positive effect on solving overlapping problems. However, this has not been verified empirically on overlapping problems. Song *et al.* [134] used a similar mechanism and identified important variables that can participate in multiple components to solve large-scale virtual network embedding problems.

Factored EA [135] is another framework with the capacity to decompose a problem into a set of lower dimensional subproblems. It can mimic CC as its special case and has the capacity to define overlapping components suitable for solving overlapping problems. FEA can be an effective method for solving the overlapping problem if the problem structure is known *a priori*. The performance of FEA remains to be checked on large-scale overlapping problems such as those proposed in the CEC'2013 large-scale benchmark suite.

The Bayesian optimization algorithm (BOA), uses Bayesian networks to represent the problem structure, which is capable of capturing overlapping components [136]. Although modeling the Bayesian network is computationally expensive, its flexibility makes it a good choice for solving overlapping problems. Empirical evidence suggests that BOA performs better than tightness detection [137]. Clustered EDAs are also among the implicit methods that improve the identification of problem structures as compared to canonical EDAs. Emmendorfer and Pozo [138] proposed a cluster-based EDA that uses the notion of concept-guided combination to better capture and exploit problem structure. This technique was shown to outperform models based on Bayesian networks.

B. Resource Allocation and the Imbalance Problem

The efficient use of computational resources is of significant importance in large-scale global optimization. The contribution of a decision variable or a group of decision variables, belonging to an underlying subfunction within the objective function, can have a varying degree of influence on the function output. This characteristic, which is often called the *imbalance* problem, can have a detrimental effect on the optimization performance if not handled properly. For example, Chuang and Chen [139] showed that the model-building process of EDAs is affected by the imbalance problem. They reported that the selected individuals based on which the probabilistic model of EDAs is updated lack the necessary information about the linkage structure of some parts of the problem. Omidvar *et al.* [140] also showed that the imbalance problem renders the round-robin optimization policy of cooperative coevolution suboptimal.

Although the imbalance issue has implications in many areas, such as multiobjective optimization [141], [142], constraint handling [122], and dynamic optimization [143], it has mostly been studied in the context of optimal component selection policy of cooperative coevolution. Contribution-based cooperative coevolution (CBCC) [140] is the first algorithm of this kind. To deal with the imbalance problem, CBCC and other contribution-aware algorithms first need to estimate the contribution of components, and second devise an exploration/exploitation policy to update the contribution of components and to optimize the influential components longer. The algorithms which will be reviewed in the rest of this section differ in the way they handle these two aspects.

1) Problem Decomposition and Quantifying Contributions: Problem decomposition and quantification of contributions are important prerequisites of an effective resource allocation policy. Under the black-box assumption, all we can observe is

TABLE I
LIST OF CONTRIBUTION-AWARE ALGORITHMS WITH A SHORT DESCRIPTION OF THEIR METHOD OF ESTIMATING THE CONTRIBUTION OF COMPONENTS AS WELL AS THEIR RESOURCE ALLOCATION POLICY

Algorithm	Estimating Contribution	Resource Allocation Policy
CBCC1 [140]	Moving average: all previous contributions $\sum \delta_k^{(t)}$	Round-robin followed by the best component (once)
CBCC2 [140]	see CBCC1	Round-robin followed by the best until stagnation
CBCC3 [144]	the most recent contribution ($\delta_k^{(t)}$)	The best component while better than the second best with occasional round-robin
CCFR [145]	Average of two most recent contributions	Equalize contributions then round-robin. It also has stagnation detection
BBCC [154]	A value function based on improvements	Multi-Arm Bandit: ϵ-greedy; SoftMax; UCB, etc
SACC1 [152]	Sensitivity analysis: Morris screening	See CBCC1
SACC2 [152]	Sensitivity analysis: Morris screening	See CBCC2
SACC3 [152]	Sensitivity analysis: Morris screening	Defining τ as a function of sensitivity analysis mechanism (Moris Screening)
OCC [124]	Delta disturbance	Replication of important decision variables in multiple components
CCAOI [148]	Normalized contributions using Gini index	The allocated resource is a function of Gini's index
FCRACC [147]	Weighted average of contributions and their st. dev	Optimizes the highest contributing component
DCCA [118]	see CBCC1	Distributed model (CPUs per component); Population size and τ are functions of # of CPUs
IMMO-CC [153]	Bi-objective: fitness and diversity	Components in the first front as determined by non-dominated sorting
CCFR [145]	Weighted average of two most recent contributions (normalized)	Equalize contributions then round-robin with stagnation detection
CBCCO [133]	Weighted average of all previous contributions	Round-robin followed by all components with contributions higher than half of the best
ICRA [142]	Weighted average of all previous contributions	Probabilistic roulette-wheel based on the contribution
F3C [149]	Most recent normalized contribution	Round-robin followed by fuzzy rule-based allocation based on contribution and diversity

the objective value and how it changes over time. In the literature, the improvement on optimizing a subset of the decision variables (a component) can have on the objective function value is taken as a unit of improvement or *contribution*. This value can be quantified for an arbitrary subset of the decision variables irrespective of whether or not they belong to an underlying subfunction. For a partially separable problem, if the ideal decomposition is known, one can study the effect of a single component on the objective value by freezing all other components. However, this may not be possible for an overlapping problem (see Section VI-A), where an optimal decomposition may not be known. It is therefore apparent that problem decomposition and estimation of contributions are closely interconnected.

Most algorithms define the contribution of a component to be a function of the improvement it makes on the overall objective value when it is optimized for a predetermined number of iterations while all other components are kept constant. Let $\delta_k^{(t)}$ be the improvement we get at time t when the kth component is optimized for τ iterations. Based on this definition, the two original versions of the CBCC algorithms (called CBCC1 and CBCC2) define the contribution of the kth components to be $(1/T)\sum_{t=1}^{T}\delta_k^{(t)}$, i.e., the average of all previous improvements up to the current iteration. Due to the nonstationary nature of the underlying distribution of the contributions, some algorithms defined the contribution as a moving average over the last L iterations. CBCC3 [144] used the extreme case of $L = 1$, while in the case of CCFR [145] $L = 2$. CCFR2 [146] improves upon CCFR by averaging the improvements per function evaluations to account for unequal subpopulations. CCFR2 [146] and some other algorithms [133], [142] exponentially decay the effect of historical contributions. Ren *et al.* [147] defined the contribution as a function of both $\delta_k^{(t)}$ and the standard deviation across all components. Some authors suggested various normalization of $\delta_k^{(t)}$ [118], [133], [142], [148], [149] as the contribution of a component.

In addition to $\delta_k^{(t)}$ defined previously, other measures of contributions have also been proposed. Global sensitivity analysis techniques, such as Morris screening is used in several works as the contribution measure for individual variables as well as components [150]–[152]. Delta disturbance is a perturbation method proposed to measure the contribution of individual variables and finding the most influential variables for further optimization. Using the plain fitness of a component as the contribution of a component has also been suggested [153].

2) Resource Allocation Policies: A simple allocation policy, the variations of which are used by many algorithms, is to complement the round-robin policy of canonical CC by one or more iterations of optimizing the highest contributing component (as measured by the methods outlined in Section VI-B1 and Table I). CBCC1 [140] is the most conservative approaches in which the round robin is followed by only one episode of optimizing the highest contributing component (also employed by SACC1 [152]. CBCC2 [140] is greedier and exploits the highest contributing component until no improvement is observed (also employed by SACC2 [152]). This is shown to be an unstable policy because the contribution of initially best component may not remain the best for the rest of a run. CBCC3 [144] addresses this issue by optimizing the highest contributing component until its contribution drops below the second best. This has the effect of equalizing contributions. CBCC3 also randomly enters an exploration phase where all components get a chance of updating their contributions. CCFR [145] and CCFR2 [146] use a similar equalization strategy and includes a stagnation detection mechanism to avoid optimizing stagnant components. FCRACC [147] is similar to CBCC3 in which it exploits the best component, but its method of quantifying contributions differs (see Section VI-B1). Meselhi *et al.* [149] used round robin followed by a fuzzy rule-based allocation based on the contribution and population diversity. Shen *et al.* [142] used a roulette-wheel selection mechanism based on the contributions. Jia *et al.* [133] optimized all the components whose contribution is more than half the best contribution. Although not specifically designed to address the imbalance problem, overlapped CC [124] replicates influential decision variables in more than one component, which has the effect of optimizing influential variables more often.

In addition to the heuristics described above, some algorithms define the resource allocation policy as a function of the estimated contributions. SACC3 [152] determines the number of times a component is optimized (τ) as a function of its effect as measured by Morris screening. CCAOI [148] normalizes the contributions according to the Gini index and allocates the computational resources accordingly. DCCA [118] uses a distributed model and allocates more CPU instances to better contributing components. The population size and τ are then defined to be functions of the number of CPUs assigned to a component. Bandit-based CC (BBCC) [154] uses multiarm bandit approaches, such as ϵ-greedy, SoftMax, or upper confidence bound (UCB) to select the components based on their contributions aiming at balancing exploration and exploitation in a more systematic way.

3) Component Selection as Multiarmed Bandit Problem: Despite being effective in outperforming the canonical CC, the contribution-aware algorithms discussed so far are based on a set of heuristics derived form empirical observations with minimal theoretical basis. Some authors proposed that the component selection policy of CC can be treated as a multiarmed bandit (MAB) problem [154], [155]. Among them, Kazimipour *et al.* [154] defined and mapped the building blocks of a contribution-aware CC into a MAB framework. This framework, BBCC, has the flexibility to mimic the previously described algorithms as a special case. BBCC treats the contribution of a component as the utility or the value function in reinforcement learning. This estimated contribution or long-term utility is defined to be a function of immediate improvements or rewards measured by quantities, such as $\delta_k^{(t)}$. Given this interpretation, a wide range of contribution estimators, such as moving average (simple, weighted, or exponential), rank-based, or hybrid estimators can be used. The component selection policy is also responsible for balancing between exploration and exploitation, which can be done using a wide range of existing selectors, such as ϵ-greedy, ϵ-first, GreedyMix, LeastTaken, SoftMax, UCB, and other similar selectors widely used in the multiarm bandit and reinforcement learning literature. The simplest instance of BBCC with a normalized $\delta_k^{(t)}$, and simple averaging as the contribution estimator, and ϵ-greedy has shown to outperform all CBCC family of algorithms, as well as CCFR and MOFBVE on the CEC'2013 large-scale benchmark suite.

C. Multiobjective Optimization

Multiobjective optimization problems are prevalent in a wide range of application areas [156]. As the name implies, these problems have two or more conflicting objectives causing them to have multiple tradeoff solutions known as the Pareto-optimal solutions. The scalability of multiobjective problems can be studied in either the objective space [157] or the decision space. The former refers to the effect of the number of objective functions on the performance of the algorithms [157], while the latter is concerned with the scalability of each objective function with respect to its number of decision variables. In this section, we address the scalability of multiobjective problems in the decision space, which has attracted attention in the last decade [158]–[160].

Large-scale multiobjective approaches can be seen in four major categories.

1) Problem decomposition where the aim is to break the problem into a set of lower dimensional subproblems in the decision space.
2) Problem transformation where the aim is to reformulate the original problem into a single lower dimensional problem.
3) Operator design where the aim is to devise more efficient solution generation mechanisms.
4) Problem reduction and intrinsic dimensions where the aim is to find a lower dimensional latent space embedded in a higher dimensional sparse space.

Problem decomposition approaches, not to be confused with scalarization techniques, such as Tchebychev [161], operate in the decision space and aim at forming a set of smaller and more manageable subproblems. Such decompositions are often done by considering the interaction structure of the decision variables [115] or by grouping the variables based on their effect in the objective space, i.e., those pertaining to the convergence of solutions toward the Pareto-optimal front, and those pertaining to diversity of the solutions on the Pareto-optimal front [162], [163]. Problem decomposition by means of variable interaction detection is a challenging task in large-scale multiobjective optimization for the following major reasons.

1) The lack of consistent partially separable grouping across all objectives.
2) The effect of problem formulation on variable interaction [164]. For example, the use of scalarization techniques may result in an sparse but overlapping interaction structure due to shared decision variables among several objectives (see Section VI-A for more information about overlapping problems).
3) The cost of variable interaction detection for several objectives.

Cooperative coevolution has been used with several decomposition strategies, such as random grouping [165], [166], multilevel random grouping [167], differential grouping [114], [168], and monotonicity detection [57], [162] to solve large-scale multiobjective problems. In some cases, several variable decomposition methods are combined [142] or completely new ones are proposed [169]. For instance, Ma *et al.* [170] proposed to use a convergence relevance degree to decompose the problem and Wang *et al.* [169] used a tensor canonical polyadic decomposition to divide the d-order tensor of decision variables into a set of uncorrelated components. Some other decomposition techniques [114], [162], [163], [168], [171] divide the decision variables based on their dominant role in the optimization problems: convergence of solutions on the Pareto-optimal front and diversity of solution on the Pareto-optimal front. In such techniques, it is customary to further divide the convergence matrix using variable interaction methods described before.

An alternative to problem decomposition is to *transform* the original large-scale problem into a single low-dimensional problem. This transformation can be implemented

TABLE II
CATEGORIZATION OF LARGE-SCALE ALGORITHMS WITH RESPECT TO THEIR STRATEGY OF HANDLING MULTIPLE OBJECTIVES

Type	Reference
Dominance	Cheng et al. [57], Shen et al. [142], Zhang et al. [163], Antonio and Coello [165], Wang et al. [169], Chen et al. [171], Liu et al. [176], Tian et al. [177, 178, 179, 180], Yin et al. [183], Qin et al. [185], Xiao et al. [188], Zhang et al. [189, 190], Ho et al. [191], Gong et al. [192]
Decomposition	Tian et al. [42], Liu et al. [89], Su et al. [113], Cao et al. [114], Liu et al. [141], Antonio and Coello [166], Song et al. [167], Ma et al. [170], Yi et al. [181], Cota et al. [184], Zhang et al. [186], Wang et al. [193], Shang et al. [194]
Indicator	He et al. [173], Hong et al. [187]
Generic	Zille et al. [86], Brownlee et al. [164], Cao et al. [168], Zille et al. [175]

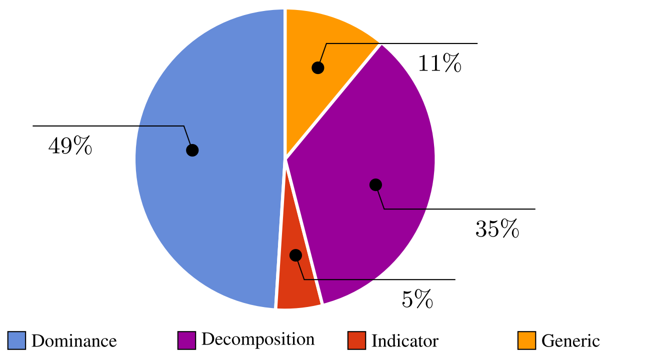

Fig. 4. Proportion of papers according to Table II using different approaches to multiobjective optimization.

using well-known methods such as PCA to obtain a lower dimensional representation of the convergence variables [89], regression analysis to represent a subset of variables as a function another subset [172], or other reformulation techniques [86], [173]. One such reformulation, which has gained attraction in recent years and has shown to outperform some state-of-the-art methods [174], is the weighted optimization framework (WOF) [86], [175]. Inspired by the notion of *adaptive weighting* [94], WOF reduces the dimensionality of the decision space by forming several groups of the decision variables, often using some variable grouping method [176], and transforming them using a given transformation function parameterized by a weight vector.

Problem reduction is another approach where the algorithm attempts to find the intrinsic dimensionality of the problem, which is often substantially lower than its nominal dimensions. Sparse multiobjective problems arise in many practical application areas where the decision value of many Pareto-optimal solutions is zero. The essence of these algorithms is to find the sparse distribution of the decision variables using techniques, such as autoencoders [177] or pattern mining [178], and devising mechanisms capable of taking the sparsity into account [179], [180].

The previous approaches stated above are all based on operating in a lower dimensional space. There are also a range of algorithm-specific ways traversing the search space more effectively. For instance, Yi *et al.* [181] studied and improved the crossover operator of NSGA-III [182]. Some studies proposed new update strategies for PSO to improve its convergence and diversity properties for large-scale multiobjective problems [42], [57], [183]. Similarly, quantum-enhanced DE [108] and variable-importance-based DE [141] have also been proposed to tackle high-dimensional multiobjective problems. Designing better local and neighborhood search mechanisms [113], [184], [185] and better solution selection mechanisms [186], [187] are other ways of improving the overall search efficiency.

Multiobjective algorithms can be categorized into dominance-based, decomposition-based, or indicator-based approaches based on how they handle the multiplicity of objective functions. Table II summarizes the large-scale multiobjective algorithms based on these three approaches. Fig. 4 shows the percentage of each approach used in the large-scale multiobjective literature. As can be seen, dominance- and decomposition-based approaches are the most dominant and indicator-based approaches are the least explored. Here, generic refers to the frameworks which are neutral to the choice of the optimizer.

D. Constraint Handling

Constraints are indispensable part of many real-world optimization problems [195]. As a result, a wide range of constraint-handling methods has been proposed for evolutionary algorithms and nature-inspired metaheuristics [196]. Despite the plethora of constraint-handling techniques, they suffer from the curse of dimensionality and very few studies have been dedicated to the topic of scalability in constrained optimization. Constraint-handling methods have the following scalability challenges.

1) High-dimensional objective and/or constraint functions.
2) The dependence between the number of constraints and the number of decision variables. For a large-scale problem, this may result in a highly constrained problem.
3) The complex problem structure due to shared decision variables among the objective function and the constraints.
4) The effect of constraint-handling method on the problem structure and variable interaction. For example, a simple penalty method can convert a partially separable problem into an overlapping one (see Section VI-A).

Constraint-handling techniques used in large-scale global optimization are mostly based on problem decomposition and variable interaction analysis. Accurate decomposition has been shown to have a significant impact on reducing the number of constraint violations [197]. Fitness difference minimization [198] and the differential grouping family [115] are two of the most widely used decomposition methods in large-scale-constrained optimization.

Sayed *et al.* [198] used a fitness difference minimization approach to analyze the interaction structure of the objective and constraint functions and decompose them into a set of smaller subproblems. Aguilar-Justo and Mezura-Montes [199] improved upon [198] and used an aggregate function of all constraint violations, rather than the individual constraints, for interaction analysis and problem decomposition. A problem of fitness difference minimization methods is the need for specifying the number or the size of components. To fix this, Aguilar-Justo *et al.* [200] proposed to evolve the best arrangement of the decision variables as well as the number of components using GA to solve large-scale-constrained problems.

Finite-difference decomposition methods (part I Section II-B) are generally more accurate than fitness difference minimization. A study shows that one such algorithm, differential grouping version 2 [115], is more robust and, therefore, better suited to complex highly constrained problems [201]. Blanchard *et al.* [202] used a variant of differential grouping, IDG [203], to form an interaction structure matrix for the objective function and the constraints. The aggregate interaction matrix of interactions is then used to decompose the objective as well as the constraints. Xu *et al.* [122] also proposed a coevolutionary algorithm based on differential grouping and used a contribution-based mechanism to allocate resources to components based on their contributions and degree of constraint violations.

In addition to problem decomposition, other approaches, such as surrogate modeling [72], memetic algorithms [201], offspring generation [204], and boundary constraints violations [69], [205] are also studied in the context of large-scale-constrained optimization. Approximation and variable reduction techniques also appear to be promising approaches for solving large-scale-constrained problems [90], [91] requiring further investigation.

E. Benchmarks and Applications

Although the ultimate goal of designing efficient algorithms is to solve real-world problems, their sheer complexity due to the entanglement of various aspects, such as constraint handling and dealing with mixed variable types, limits one's ability to conduct a focused study of a particular aspect of a problem common to a wider range of problems. Benchmark problems address this issue by capturing practical aspects, such as dimensionality, modality, structure, constraints, variable types, noise, etc., into a set of tunable well-defined functions [206]. In the context of large-scale global optimization, the IEEE CEC large-scale global optimization benchmark suites [207]–[209] are the most widely used. All the benchmarks are based on a set of base functions popular in numerical optimization [210], such as Rastrigin, Rosenbrock, Ackley, Schwefel, Sphere, Elliptical, Griewank, and many more.

The CEC'2008 large-scale suite [207] contains a set of seven functions, which is tested in 100, 500, and 1000-D dimensions. This is a very small set and lacks modularity, i.e., systematic control over how decision variables are linked. The CEC'2005 suite, though not labeled as "large scale" is scalable but used mostly in low-dimensional ($\approx$30-D) analyses. It introduces composite functions[1] through the weighted sum of a series of base functions, which can potentially cause partial interaction between decision variables. However, this is not implemented in a systematic way to give a full control over the problem structure. Herrera *et al.* [19] used a similar approach to propose a set of 19 functions used in the special issue of Soft Computing Journal on the scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems [212].

To facilitate the study of variable interaction and decomposition-based algorithms, the CEC'2010 large-scale suite [208] introduced modularity into the benchmarks, where the number of component functions and their participating decision variables are known and controllable. The benchmark contains a set of 20 1000-D functions in three categories: 1) fully separable; 2) two types of partially separable functions with and without a fully separable component; and 3) fully nonseparable. The CEC'2013 large-scale suite [209] addressed the shortcomings of its predecessor by introducing nonuniform component sizes, imbalance among the contributions of components, and overlapping functions the components of which may not be disjoint. It also includes transformations, such as symmetry breaking, ill conditioning, and local irregularities proposed in the black-box optimization benchmark (BBOB) suite [213]. Sun *et al.* [214] extended the CEC'2013 suite to make it more tunable. Other studies which paid attention to problems with overlapping components were conducted by Werth *et al.* [78] and Sayed *et al.* [215].

There are other scalable benchmarks that have been used to study various other aspects of large-scale optimization algorithms. COPS 3.0 represents a repertoire of scalable and constrained problems found in various areas of engineering and sciences. Goh *et al.* [216] used the EEG big data optimization problem and proposed a BigOpt2015 benchmark suite for large-scale multiobjective optimization. Cheng *et al.* [217] adopted the ideas proposed in [208] and [218] to propose a set of large-scale multi- and many-objective benchmark functions. Scalable benchmark functions for constrained optimization problems are very limited. Beside the constrained problems compiled by Dolan *et al.* [219], which are not widely used in the EC community, Sayed *et al.* [198], [215] also proposed a set of artificial benchmark functions for constrained problems. Recent studies also attempted to extend standard dynamic optimization benchmarks to study large-scale dynamic optimization problems [143], [220].

Applications: Benchmarks are used to ultimately help with designing and evaluating efficient algorithms for solving real-world problems. Some studies directly used real-world problem instances, in addition to artificial benchmarks, to compare algorithms. Table S-III of the supplementary material shows a set of high-dimensional optimization problems in a wide range of application areas. The problem types include real valued, integer/binary, mixed integer, and combinatorial, and about half of these include constraints. Among the approaches, divide-and-conquer methods, such as problem decomposition and coevolutionary algorithms are the most popular followed by hybrid methods (memetic algorithm, local search hybridization, and ensembles), which is consistent with our observations in part I of this survey series. Other approaches include parallelization, approximation and

[1]Composition used by [211] does not refer to composite functions generally defined as $f(g(x)) \equiv (f \circ g)(x)$. It refers to weighted sum of a set of subfunctions.

TABLE III
WINNING AND RUNNER-UP ALGORITHMS OF LARGE-SCALE GLOBAL OPTIMIZATION COMPETITIONS SINCE 2008

		Algorithms	
Year	Competition	Winner	Runner-up
2008	IEEE CEC'2008	MTS [221]	LSEDA-gl [222]
2010	IEEE CEC'2010	MA-SW-Chains [110]	EOEA [223]
2011	Soft Computing Special Issue	MOS [224]	–
2013	IEEE CEC'2013	MOS [225]	DECC-G [94]
2015	IEEE CEC'2015	MOS [225]	IHDELS [226]
2018	IEEE CEC'2018	SHADEILS [227]	LSHADE-SPA [228]
2019	IEEE CEC'2019	CC-RDG3 [126]	MPS [229]

encoding schemes, and algorithm-specific sampling and variation operator design. Curse of dimensionality is the predominant source of difficulty among these application areas and the existence of other factors, such as constraints, noisy, and nonsmooth objective functions add to the complexity. For the continuous decomposition-based approaches, noise and nonsmooth functions appear to be an obstacle to an accurate variable interaction analysis. In the case of combinatorial problems, finding an effective decomposition is a major challenge in its own right. A challenge for approximation and encoding schemes is model resolution or granularity which mediates between the accuracy of the model and its complexity. Among the constrained problems, handling of infeasible solutions and the interplay between dimensionality and the number of constraints are two of the most important challenges.

Large-Scale Global Optimization Competitions: In this section, we review the results of large-scale global competitions since 2008 when the first IEEE CEC competition on large-scale global optimization was held. Table III lists the winners and runner ups. Based on the approaches to large-scale global optimization we outlined in this article, high performing algorithms almost exclusively belong to either memetic algorithms (part I Section III) or decomposition-based algorithms (part I Section II) with memetic algorithms and local search dominating the competition.

It is interesting to note that the algorithmic philosophy of these two approaches is orthogonal and complementary to each other. The premise of memetic algorithms is to balance exploration and exploitation by means of combining global and local search operators. Whereas the main premise of decomposition methods is interaction-aware space partitioning and dimensionality reduction. Consequently, memetic algorithms lack an intrinsic mechanism for dealing with variable interactions and systematic space partitioning, and decomposition methods lack an intrinsic mechanism for balancing exploration and exploitation. It is therefore not surprising to see that the effort of hybrid algorithms is focused on proposing novel ways of balancing exploration and exploitation, and the effort of decomposition methods is centered around finding accurate interaction detection principles and effective grouping. This leaves both approaches with major blind spots. The absence of interaction detection mechanism in hybrid frameworks puts an extra burden on the exploration process by searching the regions which could have been avoided through partitioning, and reduces the efficiency of dimensionwise local search due to ignored interactions. Most decomposition-based methods also discount the role of the component optimizer in balancing exploration and exploitation. This deficiency is partially compensated by contribution-aware algorithms (see Section VI-B) which balance exploration and exploitation at the component level.

The design biases of hybrid and decomposition-based methods stated above can also be seen among the competition winners and runner ups listed in Table III. MTS [221] uses orthogonal arrays combined with a mixture of local operators. Orthogonal arrays give an expansive initial coverage of the search space, which forms the basis for the subsequent local search process. MA-SW-Chains [110], IHDELS [226], and SHADEILS [227] combine a global search algorithm with a chain of iterated local search attempts to balance exploration and exploitation. MOS [224], [225], despite being a framework capable of hybridizing any set of operators, uses a mixture of global and local operators to control exploration and exploitation. Even the algorithms, such as MPS [229] and LSEDA-gl [222], which are not memetic by definition, have explicit elements of balancing the exploration and exploitation forces. MPS [229], for instance, proposes a mechanism to disentangle the exploration and exploitation mechanisms with the aim of minimizing failed and deceptive exploration attempts and maximize the successful ones. LSEDA-gl [222] also hybridizes heavy-tailed distributions such as Lévy to promote exploration with the classic Gaussian distribution to promote exploitation.

Among the competition winners and runner ups listed in Table III, LSHADE-SPA [228] and EOEA [230] are the only hybrid algorithms that consider both exploration/exploitation balance as well as problem decomposition. On the decomposition side, however, they both fall short of using an effective variable interaction detection method. LSHADE-SPA [228] uses the outdated random grouping [94] known to perform poorly on partially separable problems (see Section II of part I) and EOEA [230] uses a grouping strategy incapable of an effective space partitioning. Due to their emphasis on accurate interaction analysis, decomposition methods are historically not systematic with their choice of component optimizer, which limits their ability in maintaining good exploration/exploitation balance within the lower dimensional subspaces. This is perhaps why hybrid algorithms are dominant in competitions [231], [232]. As a matter of fact, decomposition-based algorithms are capable of using *any* component optimizer, including memetic algorithms and other hybrids, to further improve their scalability. This is why the combination of accurate problem decomposition (RDG3 [126]) and good component optimizer (CMA-ES) resulted in superior performance by CC-RDG3 in 2019. Further evidence also suggests that several decomposition-based algorithms not present in competitions can outperform competition winners. For example, the decomposition-based algorithm proposed by Mei *et al.* [233] outperformed MA-SW-Chains [110], the winner of CEC'2010 competition on both CEC'2008 can CEC'2010 LSGO benchmark suites, and recursive differential grouping [132] outperformed MA-SW-Chains [110] and MOS [225] on the CEC'2010 and CEC'2013 LSGO benchmark suites. It is often stated that the cost of decomposition prohibits their use in large-scale settings. However, recent advances in variable interaction and grouping methods allows this to be achieved in $\mathcal{O}(n \log n)$ in the general case and $\mathcal{O}(n)$ on separable functions (see part I Section II-B for more details).

VII. Concluding Remarks

In the two parts of this survey, we reviewed a wide range of population-based metaheuristics for large-scale global optimization in six major categories: 1) problem decomposition; 2) hybridization and memetic algorithms; 3) sampling and variation operators; 4) approximation and surrogate modeling; 5) initialization methods; and 6) parallelization. We reported on the state of the art and what has been achieved over the last decade. In this section, we change perspective and try to touch upon two major issues pertaining to the future of the field.

1) Where do we stand as a field and what are the potential pitfalls and challenges hindering the progress of the field?
2) Where to go next? What are the pressing open questions and where more focus is needed?

A. Large-Scale Global Optimization: Pitfalls and Challenges

Big Picture: Despite the advances in various areas of large-scale global optimization, sometimes their relation to the bigger picture is unclear. This is partly due to the lack of a clear measure of the progress in the field and lack of clarity about its *grand challenges*. The bulk of the research in the field is currently driven by showing statistically significant improvements over existing results with minimal reference to whether these so-called *significant* results are actually meaningful in real-world settings. There is also an overemphasis of the *success* stories, rather than giving insights into *why* an algorithm *fails* on a particular problem or a class of problems. This is partly due to the departure from the scientific method in conducting research in favor of an engineering approach where a comparison with the state of the art is encouraged.

The lack of a big picture may cause the field to focus too much on *nice-to-have* incremental research rather than addressing the core issues of large-scale global optimization. This deficiency currently manifests itself in at least three forms.

1) *Emergence of New "Metaphor"-Based Algorithms:* A wide array of metaphor-based metaheuristics have been proposed in recent years. These "novel" algorithms are often a marginal variation of an existing algorithm under the disguise of new terminology. In large-scale global optimization, some works claim novelty by simply applying one such new metaheuristic to solve some standard large-scale benchmark suite. This is very detrimental to the field and "take the field of metaheuristics a step backward rather than forward" [234].
2) *Ad Hoc Improvements of Algorithms With Marginal Scientific or Practical Significance:* This type of work often present a minor variation of an existing algorithm, which *statistically* improves upon the previous results despite the magnitude of the difference being negligible for any practical purpose. As an example, applying a known parameter adaptation technique to dynamically control the parameters of a new metaheuristic algorithm falls short of addressing major challenges of the field.
3) *Theory-Practice Gap:* Given the ever growing need for scalable optimization algorithms in a wide range of application areas, the gap between theory and practice in terms of the problem sizes currently being tackled is widening. In other words, the large-scale problems being studied now using the standard benchmarks are far from the large-scale problems faced in practice.

To avoid falling prey to these defects, we need to check where we stand as a community in relation to identifying and addressing the grand challenges of the field and bridging the gap between theory and practice. This perhaps requires a separate quantitative in-depth investigation of the large body of reported results, which is outside the scope of this article.

Comparison: Despite the availability of relatively standardized and widely used benchmark suites, it is still hard to compare the reported results across a wider body of publications to be able to see the major patterns and trends. Despite some attempts to develop automated comparison tools such as the toolkit for automatic comparison of optimizers (TACOs),[2] we currently do not know the answer to questions such as the following: given a specific function or family of functions, which algorithm or class of algorithms performs the best and why? In continuous problems, especially due to the imbalance effect, the overall solution quality may seem poor, but the solution may indeed be close to the global optimum. Based on the reported results, we currently do not know how far the solutions are from the global optimum.

The large-scale global optimization competitions also acted as a venue to compare a wider range of algorithms, but their conclusions remain limited due to the absence of several state-of-the-art algorithms from the competitions. We do not know to what extent does the competition outcomes depend on the allotted number of objective function evaluations. Do the conclusions change if the algorithms are given less or more resources?

Answering some of the questions stated above can help in finding the recurring issues and bottlenecks and help with shaping the big picture and identifying the core issues of the field.

Adoption: Lack of streamlined and easy-to-use software packages make the adoption of the recent developments very difficult for practitioners or other researchers outside the field. The most recent algorithms are often sophisticated and hard to implement which is a stumbling block in the way of their wider adoption.

B. Potential Areas for Future Research

1) Synergy Between Optimization and Learning: Deep learning problems are in essence high-dimensional problems with the potential to contain millions or billions of decision variables. Although evolutionary algorithms

[2]https://tacolab.org/

have shown competitive results on high-dimensional learning problems [235], [236], research on devising population-based algorithms to tackle large-scale learning problems is scarce [237]–[242]. Population-based metaheuristics in general, and evolutionary algorithms in particular, are suited for environments that require hard exploration. As a result, they can be competitive in areas, such as neural architecture search [243], training of deep neural networks [236], and reinforcement learning [244]. In reinforcement learning, for instance, evolution strategies have shown to perform better than the policy gradient on Atari 2600 games. These methods are particularly suitable when the effective number of time steps is long, the actions have long-lasting effects, and no good value function estimator is available [244].

Conversely, machine learning algorithms can be used in conjunction with large-scale global optimization algorithms to improve their scalability. For example, machine learning can be used as a general approach to learn and discover a problem's structural information from available data [245]. The learned model can then be applied to unseen data for the purpose of classification or time-series prediction. An optimization process such as branch and bound can be modeled as a decision making process; hence, a machine learning model can be applied, to learn the most efficient and effective way [246]. This will go a long way in helping an algorithm's ability to scale to higher dimensional problems. Similarly, machine learning algorithms can be used to reduce the size of an optimization problem before being tackled. For examples, some recent work on employing machine learning techniques to learn from known instances that contain optimal solutions in order to reduce the problem first, without losing the optimal solutions [247], [248].

2) Synergy Between Metaheuristics and Classic Mathematical Programming: Several classic derivative-free optimization algorithms have been successfully used as local search operators in the context of memetic algorithms (c.f. part I Section III). Other promising areas of research at the intersection of population-based optimization algorithms and classic mathematical programming are as follows.

1) *Problem Decomposition:* In classic mathematical programming, there exist some decomposition methods, such as column generation, Bender's cut, and Dantzig–Wolfe decomposition [249]. These techniques can be effective under certain assumptions such convexity or linearity. An important question to ask is how one would combine the merits of metaheuristics with these decomposition methods to tackle real-world LSGO problems that are often nonconvex and nonlinear? As an example, machine learning has been used to learn when Dantzig–Wolf decomposition is effective on mixed-integer linear programming problems [250].
2) *Variation Operators:* Many large-scale real-world optimization problems are combinatorial by nature, e.g., either discrete, binary, or mixed types. Examples include large-scale traveling salesman problems or other graph-based optimization problems. Designing metaheuristic variation operators (such as crossover and mutation) in order to produce new solutions from the existing ones can be a significant challenge. Most existing successful methods are conventional mathematical programming methods, such as branch-and-bound and branch-and-cut methods. Hybrid methods that combine the merits of metaheuristics and exact methods (e.g., taking advantage of the "shared information" in a metaheuristic population) are a promising direction [251]–[254].

3) Exploiting Problem Structure: Exploiting the problem structure and gray-box optimization has shown to be effective ways of solving large-scale problems (part I Section 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 the 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.

4) Noise, Dynamism, and Uncertainty: The scalability of optimization algorithms in the presence of noise, dynamical changes of the landscape, and uncertainty has scarcely been studied with only few papers addressing these issues [143], [220]. These problem types pose a range of new challenges to the existing approaches to large-scale optimization presented before. For example, variable interaction analysis methods are designed based on the assumption that the objective function is noiseless. The dynamical changes of the landscape can change the structural properties of the objective function, which makes it difficult for the explicit and implicit methods to exploit these information in an efficient manner.

5) Constraint Handling: Constraints are indispensable part of most real-world optimization problems; however, very limited studies have been dedicated to the effect of problem dimensionality on constraints [198], [202] (also see Section VI-D). There is still a lack of efficient constraint handling tools to cope with high-dimensional constraint functions or the cases where the number of constraints is a function of the dimensionality of the objective function [255], [256]. In the later case, the problem may contain a large number of low-dimensional constraints. The field currently lacks scalable and controllable constrained benchmark problems either synthetic or based on real-world problems [195].

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4, 1995, pp. 1942–1948.

[2] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[3] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[4] M. S. Maučec and J. Brest, "A review of the recent use of differential evolution for large-scale global optimization: An analysis of selected algorithms on the CEC 2013 LSGO benchmark suite," *Swarm Evol. Comput.*, vol. 50, Nov. 2019, Art. no. 100428.

[5] A. W. Mohamed, A. A. Hadi, and A. K. Mohamed, "Differential evolution mutations: Taxonomy, comparison and convergence analysis," *IEEE Access*, vol. 9, pp. 68629–68662, 2021.

[6] M. Z. Ali, N. H. Awad, and P. N. Suganthan, "Multi-population differential evolution with balanced ensemble of mutation strategies for large-scale global optimization," *Appl. Soft Comput.*, vol. 33, pp. 304–327, Aug. 2015.

[7] A. Banitalebi, M. I. A. Aziz, and Z. A. Aziz, "A self-adaptive binary differential evolution algorithm for large scale binary optimization problems," *Inf. Sci.*, vols. 367–368, pp. 487–511, Nov. 2016.

[8] J.-I. Kushida, A. Hara, and T. Takahama, "Rank-based differential evolution with multiple mutation strategies for large scale global optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2015, pp. 353–360.

[9] Y. Wang, B. Li, and X. Lai, "Variance priority based cooperative co-evolution differential evolution for large scale global optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 1232–1239.

[10] A. W. Mohamed and A. S. Almazyad, "Differential evolution with novel mutation and adaptive crossover strategies for solving large scale global optimization problems," *Appl. Comput. Intell. Soft Comput.*, vol. 2017, Mar. 2017, Art. no. 7974218.

[11] A. W. Mohamed, "Solving large-scale global optimization problems using enhanced adaptive differential evolution algorithm," *Complex Intell. Syst.*, vol. 3, no. 4, pp. 205–231, Dec. 2017.

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

[13] H. Wang, Z. Wu, S. Rahnamayan, and D. Jiang, "Sequential DE enhanced by neighborhood search for large scale global optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2010, pp. 1–7.

[14] C. García-Martínez, F. J. Rodríguez, and M. Lozano, "Role differentiation and malleable mating for differential evolution: an analysis on large-scale optimisation," *Soft Comput.*, vol. 15, no. 11, pp. 2109–2126, 2011.

[15] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.

[16] Q. Yang, H.-Y. Xie, W.-N. Chen, and J. Zhang, "Multiple parents guided differential evolution for large scale optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2016, pp. 3549–3556.

[17] H. Wang, Z. Wu, and S. Rahnamayan, "Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems," *Soft Comput.*, vol. 15, no. 11, pp. 2127–2140, 2011.

[18] S. Rahnamayan, H. R. Tizhoosh, and M. Salama, "Opposition-based differential evolution algorithms," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2006, pp. 2010–2017.

[19] F. Herrera, M. Lozano, and D. Molina, "Test suite for the special issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems," 2009. [Online]. Available: http://150.214.190.154/sites/default/files/files/TematicWebSites/EAMHCO/fu nctions1-19.pdf

[20] H. Hiba, S. Mahdavi, and S. Rahnamayan, "Differential evolution with center-based mutation for large-scale optimization," in *Proc. IEEE Symp. Series Comput. Intell.*, 2017, pp. 1–8.

[21] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer, "High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction," in *Proc. IEEE Congr. Evol. Comput. (IEEE World Congr. Comput. Intell.)*, 2008, pp. 2032–2039.

[22] H. Wang, S. Rahnamayan, and Z. Wu, "Parallel differential evolution with self-adapting control parameters and generalized opposition-based learning for solving high-dimensional optimization problems," *J. Parallel Distrib. Comput.*, vol. 73, no. 1, pp. 62–73, 2013.

[23] J. Brest, A. Zamuda, I. Fister, and M. S. Maučec, "Large scale global optimization using self-adaptive differential evolution algorithm," in *Proc. IEEE Congr. Evol. Comput.*, 2010, pp. 1–8.

[24] M. Weber, F. Neri, and V. Tirronen, "Shuffle or update parallel differential evolution for large-scale optimization," *Soft Comput.*, vol. 15, no. 11, pp. 2089–2107, 2011.

[25] A. Zamuda, J. Brest, B. Boskovic, and V. Zumer, "Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution," in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 3718–3725.

[26] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2, 2005, pp. 1785–1791.

[27] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *Proc. IEEE Congr. Evol. Comput. (IEEE World Congr. Comput. Intell.)*, 2008, pp. 1110–1116.

[28] Z. Yang, K. Tang, and X. Yao, "Scalability of generalized adaptive differential evolution for large-scale continuous optimization," *Soft Comput.*, vol. 15, no. 11, pp. 2141–2155, 2011.

[29] T. Takahama and S. Sakai, "Large scale optimization by differential evolution with landscape modality detection and a diversity archive," in *Proc. IEEE Congr. Evol. Comput.*, 2012, pp. 1–8.

[30] H. Wang, S. Rahnamayan, and Z. Wu, "Adaptive differential evolution with variable population size for solving high-dimensional problems," in *Proc. IEEE Congr. Evol. Comput.*, 2011, pp. 2626–2632.

[31] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2014, pp. 1658–1665.

[32] J. Lampinen and I. Zelinka, "On stagnation of the differential evolution algorithm," in *Proc. MENDEL*, 2000, pp. 76–83.

[33] C. Segura, C. A. Coello Coello, and A. G. Hernández-Díaz, "Improving the vector generation strategy of differential evolution for large-scale optimization," *Inf. Sci.*, vol. 323, pp. 106–129, Dec. 2015.

[34] Y.-F. Ge, W.-J. Yu, and J. Zhang, "Diversity-based multi-population differential evolution for large-scale optimization," in *Proc. Genet. Evol. Comput. Conf.*, 2016, pp. 31–32.

[35] Y.-F. Ge *et al.*, "Distributed differential evolution based on adaptive mergence and split for large-scale optimization," *IEEE Trans. Cybern.*, vol. 48, no. 7, pp. 2166–2180, Jul. 2018.

[36] K. E. Parsopoulos, "Cooperative micro-differential evolution for high-dimensional problems," in *Proc. Genet. Evol. Comput. Conf.*, 2009, pp. 531–538.

[37] E. H. Houssein, A. G. Gad, K. Hussain, and P. N. Suganthan, "Major advances in particle swarm optimization: Theory, analysis, and application," *Swarm Evol. Comput.*, vol. 63, Jun. 2021, Art. no. 100868.

[38] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proc. Congr. Evol. Comput.*, vol. 2, 2004, pp. 1980–1987.

[39] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.

[40] N. Lynn and P. N. Suganthan, "Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation," *Swarm Evol. Comput.*, vol. 24, pp. 11–24, Oct. 2015.

[41] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 191–204, Feb. 2015.

[42] Y. Tian, X. Zheng, X. Zhang, and Y. Jin, "Efficient large-scale multiobjective optimization based on a competitive swarm optimizer," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3696–3708, Aug. 2020.

[43] E. Naderi, H. Narimani, M. Fathi, and M. R. Narimani, "A novel fuzzy adaptive configuration of particle swarm optimization to solve large-scale optimal reactive power dispatch," *Appl. Soft Comput.*, vol. 53, pp. 441–456, Apr. 2017.

[44] R.-L. Tang, Z. Wu, and Y.-J. Fang, "Adaptive multi-context cooperatively coevolving particle swarm optimization for large-scale problems," *Soft Comput.*, vol. 21, no. 16, pp. 4735–4754, 2016.

[45] M. Pluhacek, R. Senkerik, and I. Zelinka, "Investigation on the performance of a new multiple choice strategy for PSO algorithm in the task of large scale optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, 2013, pp. 2007–2011.

[46] N. Lynn, M. Z. Ali, and P. N. Suganthan, "Population topologies for particle swarm optimization and differential evolution," *Swarm Evol. Comput.*, vol. 39, pp. 24–35, Apr. 2018.

[47] J. Fan, J. Wang, and M. Han, "Cooperative coevolution for large-scale optimization based on kernel fuzzy clustering and variable trust region methods," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 4, pp. 829–839, Aug. 2014.

[48] Q. Zhang, H. Cheng, Z. Ye, and Z. Wang, "A competitive swarm optimizer integrated with cauchy and Gaussian mutation for large scale optimization," in *Proc. Chin. Control Conf.*, 2017, pp. 9829–9834.

[49] Q. Yang *et al.*, "A distributed swarm optimizer with adaptive communication for large-scale optimization," *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 3393–3408, Jul. 2020.

[50] Z.-J. Wang *et al.*, "Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2715–2729, Jun. 2020.

[51] Z.-J. Wang, Z.-H. Zhan, S. Kwong, H. Jin, and J. Zhang, "Adaptive granularity learning distributed particle swarm optimization for large-scale optimization," *IEEE Trans. Cybern.*, vol. 51, no. 3, pp. 1175–1188, Mar. 2021.

[52] M. A. Arasomwan and A. O. Adewumi, "An adaptive velocity particle swarm optimization for high-dimensional function optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2013, pp. 2352–2359.

[53] R. Cheng and Y. Jin, "A social learning particle swarm optimization algorithm for scalable optimization," *Inf. Sci.*, vol. 291, pp. 43–60, Jan. 2015.

[54] J.-R. Jian, Z.-G. Chen, Z.-H. Zhan, and J. Zhang, "Region encoding helps evolutionary computation evolve faster: A new solution encoding scheme in particle swarm for large-scale optimization," *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 779–793, Aug. 2021.

[55] Q. Yang, W.-N. Chen, J. Da Deng, Y. Li, T. Gu, and J. Zhang, "A level-based learning swarm optimizer for large-scale optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 4, pp. 578–594, Aug. 2018.

[56] H. Huang, L. Lv, S. Ye, and Z. Hao, "Particle swarm optimization with convergence speed controller for large-scale numerical optimization," *Soft Comput.*, vol. 23, no. 12, pp. 4421–4437, Jun. 2019.

[57] S. Cheng, H. Zhan, H. Yao, H. Fan, and Y. Liu, "Large-scale many-objective particle swarm optimizer with fast convergence based on Alpha-stable mutation and Logistic function," *Appl. Soft Comput.*, vol. 99, Feb. 2021, Art. no. 106947.

[58] D. Li, W. Guo, A. Lerch, Y. Li, L. Wang, and Q. Wu, "An adaptive particle swarm optimizer with decoupled exploration and exploitation for large scale optimization," *Swarm Evol. Comput.*, vol. 60, Feb. 2021, Art. no. 100789.

[59] Y. Xue, T. Tang, W. Pang, and A. X. Liu, "Self-adaptive parameter and strategy based particle swarm optimization for large-scale feature selection problems with multiple classifiers," *Appl. Soft Comput.*, vol. 88, Mar. 2020, Art. no. 106031.

[60] S.-T. Hsieh, T.-Y. Sun, C.-C. Liu, and S.-J. Tsai, "Solving large scale global optimization using improved particle swarm optimizer," in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 1777–1784.

[61] 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 Comput.*, vol. 15, no. 11, pp. 2233–2255, 2011.

[62] M. A. M. De Oca, K. Van den Enden, and T. Stützle, "Incremental particle swarm-guided local search for continuous optimization," in *Proc. Int. Workshop Hybrid Metaheuristics*, 2008, pp. 72–86.

[63] J. Garcí-Nieto and E. Alba, "Restart particle swarm optimization with velocity modulation: A scalability test," *Soft Comput.*, vol. 15, no. 11, pp. 2221–2232, 2011.

[64] S. Cheng, Y. Shi, and Q. Qin, "Dynamical exploitation space reduction in particle swarm optimization for solving large scale problems," in *Proc. IEEE Congr. Evol. Comput.*, 2012, pp. 1–8.

[65] J. Zhou, W. Fang, X. Wu, J. Sun, and S. Cheng, "An opposition-based learning competitive particle swarm optimizer," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2016, pp. 515–521.

[66] T. Hendtlass, "Particle swarm optimisation and high dimensional problem spaces," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 1988–1994.

[67] T. Korenaga, T. Hatanaka, and K. Uosaki, "Performance improvement of particle swarm optimization for high-dimensional function optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2007, pp. 3288–3293.

[68] W. Chu, X. Gao, and S. Sorooshian, "A new evolutionary search strategy for global optimization of high-dimensional problems," *Inf. Sci.*, vol. 181, no. 22, pp. 4909–4927, 2011.

[69] W. Chu, X. Gao, and S. Sorooshian, "Fortify particle swam optimizer (PSO) with principal components analysis: A case study in improving bound-handling for optimizing high-dimensional and complex problems," in *Proc. IEEE Congr. Evol. Comput.*, 2011, pp. 1644–1648.

[70] Q. Zhang, W. Liu, X. Meng, B. Yang, and A. V. Vasilakos, "Vector coevolving particle swarm optimization algorithm," *Inf. Sci.*, vols. 394–395, pp. 273–298, Jul. 2017.

[71] 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 *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 3845–3852.

[72] R. G. Regis, "Evolutionary programming for high-dimensional constrained expensive black-box optimization using radial basis functions," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 326–347, Jun. 2014.

[73] P. Yang, K. Tang, and X. Yao, "Turning high-dimensional optimization into computationally expensive optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 143–156, Feb. 2018.

[74] S. Mahdavi, M. E. Shiri, and S. Rahnamayan, "Cooperative co-evolution with a new decomposition method for large-scale optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2014, pp. 1285–1292.

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

[76] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 378–393, Jun. 2014.

[77] B. Pang, Z. Ren, Y. Liang, and A. Chen, "Enhancing cooperative coevolution for large scale optimization by adaptively constructing surrogate models," 2018, *arXiv:1803.00906*.

[78] B. Werth, E. Pitzer, and M. Affenzeller, "Enabling high-dimensional surrogate-assisted optimization by using sliding windows," in *Proc. Genet. Evol. Comput. Conf. Companion*, 2017, pp. 1630–1637.

[79] C. Wang and J.-H. Gao, "A differential evolution algorithm with cooperative coevolutionary selection operation for high-dimensional optimization," *Optim. Lett.*, vol. 8, no. 2, pp. 477–492, 2014.

[80] I. De Falco, A. D. Cioppa, and G. A. Trunfio, "Large scale optimization of computationally expensive functions: An approach based on parallel cooperative coevolution and fitness metamodeling," in *Proc. Genet. Evol. Comput. Conf. Companion*, 2017, pp. 1788–1795.

[81] Z. Ren, B. Pang, Y. Liang, A. Chen, and Y. Zhang, "Surrogate model assisted cooperative coevolution for large scale optimization," 2018, *arXiv:1802.09746*.

[82] C. Sun, Y. Jin, J. Ding, and J. Zeng, "Fitness estimation strategy assisted competitive swarm optimizer for high dimensional expensive problems," in *Proc. Genet. Evol. Comput. Conf.*, 2016, pp. 1277–1278.

[83] C. Sun, Y. Jin, R. Cheng, J. Ding, and J. Zeng, "Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 644–660, Aug. 2017.

[84] Z. Yang, B. Sendhoff, K. Tang, and X. Yao, "Target shape design optimization by evolving B-splines with cooperative coevolution," *Appl. Soft Comput.*, vol. 48, pp. 672–682, Nov. 2016.

[85] A. Tiwari, R. Roy, G. Jared, and O. Munaux, "Interaction and multi-objective optimisation," in *Proc. Genet. Evol. Comput. Conf.*, 2001, pp. 671–678.

[86] H. Zille, H. Ishibuchi, S. Mostaghim, and Y. Nojima, "A framework for large-scale multiobjective optimization based on problem transformation," *IEEE Trans. Evol. Comput.*, vol. 22, no. 2, pp. 260–275, Apr. 2018.

[87] A. Kabán, J. Bootkrajang, and R. J. Durrant, "Toward large-scale continuous EDA: A random matrix theory perspective," *Evol. Comput.*, vol. 24, no. 2, pp. 255–291, Jun. 2016.

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

[89] R. Liu, R. Ren, J. Liu, and J. Liu, "A clustering and dimensionality reduction based evolutionary algorithm for large-scale multi-objective problems," *Appl. Soft Comput.*, vol. 89, Apr. 2020, Art. no. 106120.

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

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

[92] M. L. Sanyang and A. Kabán, "REMEDA: Random embedding EDA for optimising functions with intrinsic dimension," in *Parallel Problem Solving From Nature*. Heidelberg, Germany: Springer, 2016, pp. 859–868.

[93] S. Wang, J. Liu, and Y. Jin, "Surrogate-assisted robust optimization of large-scale networks based on graph embedding," *IEEE Trans. Evol. Comput.*, vol. 24, no. 4, pp. 735–749, Aug. 2020.

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

[95] B. Kazimipour, X. Li, and A. K. Qin, "A review of population initialization techniques for evolutionary algorithms," in *Proc. IEEE Congr. Evol. Comput.*, 2014, pp. 2585–2592.

[96] S. Mahdavi, S. Rahnamayan, and K. Deb, "Center-based initialization of cooperative co-evolutionary algorithm for large-scale optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2016, pp. 3557–3565.

[97] B. Kazimipour, X. Li, and A. K. Qin, "Initialization methods for large scale global optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2013, pp. 2750–2757.

[98] B. Kazimipour, X. Li, and A. K. Qin, "Effects of population initialization on differential evolution for large scale optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2014, pp. 2404–2411.

[99] B. Kazimipour, X. Li, and A. Qin, "Why advanced population initialization techniques perform poorly in high dimension?" in *Simulated Evolution and Learning* (LNCS 8886). Heidelberg, Germany: Springer Int., 2014, pp. 479–490.

[100] E. Segredo, B. Paechter, C. Segura, and C. I. González-Vila, "On the comparison of initialisation strategies in differential evolution for large scale optimisation," *Optim. Lett.*, vol. 12, no. 1, pp. 221–234, 2018.

[101] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is 'nearest neighbor' meaningful?" in *Proc. Int. Conf. Database Theory*, 1999, pp. 217–235.

[102] R. J. Durrant and A. Kabán, "When is 'nearest neighbour' meaningful: A converse theorem and implications," *J. Complex.*, vol. 25, no. 4, pp. 385–397, 2009.

[103] E. Cantú-Paz and D. E. Goldberg, "On the scalability of parallel genetic algorithms," *Evol. Comput.*, vol. 7, no. 4, pp. 429–449, Dec. 1999.

[104] M. Munetomo, N. Murao, and K. Akama, "Empirical investigations on parallelized linkage identification," in *Parallel Problem Solving From Nature*. Heidelberg, Germany: Springer, 2004, pp. 322–331.

[105] A. Mendiburu, J. A. Lozano, and J. Miguel-Alonso, "Parallel implementation of EDAs based on probabilistic graphical models," *IEEE Trans. Evol. Comput.*, vol. 9, no. 4, pp. 406–423, Aug. 2005.

[106] S. Iturriaga and S. Nesmachnow, "Solving very large optimization problems (up to one billion variables) with a parallel evolutionary algorithm in CPU and GPU," in *Proc. 7th Int. Conf. P2P Parallel Grid Cloud Internet Comput.*, 2012, pp. 267–272.

[107] Q. Duan, L. Sun, and Y. Shi, "Spark clustering computing platform based parallel particle swarm optimizers for computationally expensive global optimization," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 2018, pp. 424–435.

[108] B. Cao, S. Fan, J. Zhao, P. Yang, K. Muhammad, and M. Tanveer, "Quantum-enhanced multiobjective large-scale optimization via parallelism," *Swarm Evol. Comput.*, vol. 57, Sep. 2020, Art. no. 100697.

[109] M. Lastra, D. Molina, and J. M. Benítez, "A high performance memetic algorithm for extremely high-dimensional problems," *Inf. Sci.*, vol. 293, pp. 35–58, Feb. 2015.

[110] D. Molina, M. Lozano, and F. Herrera, "MA-SW-Chains: Memetic algorithm based on local search chains for large scale continuous global optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2010, pp. 3153–3160.

[111] A. Cano and C. García-Martínez, "100 million dimensions large-scale global optimization using distributed GPU computing," in *Proc. IEEE Congr. Evol. Comput.*, 2016, pp. 3566–3573.

[112] A. Cano, C. García-Martínez, and S. Ventura, "Extremely high-dimensional optimization with mapreduce: Scaling functions and algorithm," *Inf. Sci.*, vols. 415–416, pp. 110–127, Nov. 2017.

[113] Y. Su, K. Zhou, X. Zhang, R. Cheng, and C. Zheng, "A parallel multi-objective evolutionary algorithm for community detection in large-scale complex networks," *Inf. Sci.*, vol. 576, pp. 374–392, Oct. 2021.

[114] B. Cao, J. Zhao, Z. Lv, and X. Liu, "A distributed parallel cooperative coevolutionary multiobjective evolutionary algorithm for large-scale optimization," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 2030–2038, Aug. 2017.

[115] 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 Trans. Evol. Comput.*, vol. 21, no. 6, pp. 929–942, Dec. 2017.

[116] P. Yang, K. Tang, and X. Yao, "A parallel divide-and-conquer based evolutionary algorithm for large-scale optimization," 2018, *arXiv:1812.02500*.

[117] G. Roy, H. Lee, J. L. Welch, Y. Zhao, V. Pandey, and D. Thurston, "A distributed pool architecture for genetic algorithms," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 1177–1184.

[118] Y.-H. Jia *et al.*, "Distributed cooperative co-evolution with adaptive computing resource allocation for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 2, pp. 188–202, Apr. 2019.

[119] J. R. R. A. Martins and A. B. Lambe, "Multidisciplinary design optimization: A survey of architectures," *AIAA J.*, vol. 51, no. 9, pp. 2049–2075, 2013.

[120] A. Yassine and D. Braha, "Complex concurrent engineering and the design structure matrix method," *Concurrent Eng.*, vol. 11, no. 3, pp. 165–176, 2003.

[121] K. Li, M. N. Omidvar, K. Deb, and X. Yao, "Variable interaction in multi-objective optimization problems," in *Parallel Problem Solving From Nature*. Cham, Switzerland: Springer Int., 2016, pp. 399–409.

[122] P. Xu, W. Luo, X. Lin, J. Zhang, Y. Qiao, and X. Wang, "Constraint-objective cooperative coevolution for large-scale constrained optimization," *ACM Trans. Evol. Learn. Optim.*, vol. 1, no. 3, pp. 1–26, Aug. 2021.

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

[124] A. Song, W.-N. Chen, P.-T. Luo, Y.-J. Gong, and J. Zhang, "Overlapped cooperative co-evolution for large scale optimization," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 2017, pp. 3689–3694.

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

[126] Y. Sun, X. Li, A. Ernst, and M. N. Omidvar, "Decomposition for large-scale optimization problems with overlapping components," in *Proc. IEEE Congr. Evol. Comput.*, 2019, pp. 326–333.

[127] L. Li, W. Fang, Y. Mei, and Q. Wang, "Cooperative coevolution for large-scale global optimization based on fuzzy decomposition," *Soft Comput.*, vol. 25, no. 5, pp. 3593–3608, Mar. 2021.

[128] D. Thierens, "The linkage tree genetic algorithm," in *Parallel Problem Solving From Nature*. Heidelberg, Germany: Springer, 2010, pp. 264–273.

[129] P. A. N. Bosman and D. Thierens, "More concise and robust linkage learning by filtering and combining linkage hierarchies," in *Proc. Genet. Evol. Comput. Conf.*, 2013, pp. 359–366.

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

[131] T.-L. Yu, K. Sastry, and D. E. Goldberg, "Linkage learning, overlapping building blocks, and systematic strategy for scalable recombination," in *Proc. Genet. Evol. Comput. Conf.*, 2005, pp. 1217–1224.

[132] Y. Sun, M. Kirley, and S. K. Halgamuge, "A recursive decomposition method for large scale continuous optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 5, pp. 647–661, Oct. 2018.

[133] Y.-H. Jia, Y. Mei, and M. Zhang, "Contribution-based cooperative co-evolution for nonseparable large-scale problems with overlapping subcomponents," *IEEE Trans. Cybern.*, early access, Oct. 29, 2020, doi: 10.1109/TCYB.2020.3025577.

[134] A. Song, W.-N. Chen, Y.-J. Gong, X. Luo, and J. Zhang, "A divide-and-conquer evolutionary algorithm for large-scale virtual network embedding," *IEEE Trans. Evol. Comput.*, vol. 24, no. 3, pp. 566–580, Jun. 2020.

[135] S. Strasser, J. Sheppard, N. Fortier, and R. Goodman, "Factored evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 281–293, Apr. 2017.

[136] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The Bayesian optimization algorithm," in *Proc. 1st Annu. Conf. Genet. Evol. Comput. Vol. 1*, 1999, pp. 525–532.

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

[138] L. R. Emmendorfer and A. T. R. Pozo, "Effective linkage learning using low-order statistics and clustering," *IEEE Trans. Evol. Comput.*, vol. 13, no. 6, pp. 1233–1246, Dec. 2009.

[139] C.-Y. Chuang and Y.-P. Chen, "Sensibility of linkage information and effectiveness of estimated distributions," *Evol. Comput.*, vol. 18, no. 4, pp. 547–579, 2010.

[140] M. N. Omidvar, X. Li, and X. Yao, "Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms," in *Proc. Genet. Evol. Comput. Conf.*, 2011, pp. 1115–1122.

[141] S. Liu, Q. Lin, Y. Tian, and K. C. Tan, "A variable importance-based differential evolution for large-scale multiobjective optimization," *IEEE Trans. Cybern.*, early access, Aug. 18, 2021, doi: 10.1109/TCYB.2021.3098186.

[142] X. Shen, Y. Guo, and A. Li, "Cooperative coevolution with an improved resource allocation for large-scale multi-objective software project scheduling," *Appl. Soft Comput.*, vol. 88, Mar. 2020, Art. no. 106059.

[143] D. Yazdani, M. N. Omidvar, J. Branke, T. T. Nguyen, and X. Yao, "Scaling up dynamic optimization problems: A divide-and-conquer approach," *IEEE Trans. Evol. Comput.*, vol. 24, no. 1, pp. 1–15, Feb. 2020.

[144] M. N. Omidvar, B. Kazimipour, X. Li, and X. Yao, "CBCC3—A contribution-based cooperative co-evolutionary algorithm with improved exploration/exploitation balance," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Vancouver, BC, Canada, 2016, pp. 3541–3548.

[145] M. Yang *et al.*, "Efficient resource allocation in cooperative co-evolution for large-scale global optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 493–505, Aug. 2017.

[146] M. Yang, A. Zhou, C. Li, J. Guan, and X. Yan, "CCFR2: A more efficient cooperative co-evolutionary framework for large-scale global optimization," *Inf. Sci.*, vol. 512, pp. 64–79, Feb. 2020.

[147] Z. Ren, Y. Liang, A. Zhang, Y. Yang, Z. Feng, and L. Wang, "Boosting cooperative coevolution for large scale optimization with a fine-grained computation resource allocation strategy," 2018, *arXiv:1802.09703*.

[148] G. A. Trunfio, "Adaptation in cooperative coevolutionary optimization," in *Adaptation and Hybridization in Computational Intelligence*. Cham, Switzerland: Springer, 2015, pp. 91–109.

[149] M. A. Meselhi, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Contribution based co-evolutionary algorithm for large-scale optimization problems," *IEEE Access*, vol. 8, pp. 203369–203381, 2020.

[150] S. Mahdavi, S. Rahnamayan, and M. E. Shiri, "Multilevel framework for large-scale global optimization," *Soft Comput.*, vol. 21, pp. 4111–4140, Feb. 2016.

[151] S. Mahdavi, S. Rahnamayan, and M. E. Shiri, "Incremental cooperative coevolution for large-scale global optimization," *Soft Comput.*, vol. 22, pp. 2045–2064, Dec. 2016.

[152] S. Mahdavi, S. Rahnamayan, and M. E. Shiri, "Cooperative co-evolution with sensitivity analysis-based budget assignment strategy for large-scale global optimization," *Appl. Intell.*, vol. 47, no. 3, pp. 888–913, 2017.

[153] X. Peng and Y. Wu, "Large-scale cooperative co-evolution with bi-objective selection based imbalanced multi-modal optimization," in *Proc. IEEE Congr. Evol. Comput.*, Donostia, Spain, 2017, pp. 1527–1532.

[154] B. Kazimipour, M. N. Omidvar, A. K. Qin, X. Li, and X. Yao, "Bandit-based cooperative coevolution for tackling contribution imbalance in large-scale optimization problems," *Appl. Soft Comput.*, vol. 76, pp. 265–281, Mar. 2019.

[155] F.-M. D. Rainville, M. Sebag, C. Gagné, M. Schoenauer, and D. Laurendeau, "Sustainable cooperative coevolution with a multi-armed bandit," in *Proc. 15th Annu. Conf. Genet. Evol. Comput.*, 2013, pp. 1517–1524.

[156] C. A. Coello Coello and G. B. Lamont, *Applications of Multi-Objective Evolutionary Algorithms*, vol. 1. Singapore: World Sci., 2004.

[157] B. Li, J. Li, K. Tang, and X. Yao, "Many-objective evolutionary algorithms: A survey," *ACM Comput. Surv.*, vol. 48, no. 1, p. 13, 2015.

[158] J. J. Durillo, A. J. Nebro, C. A. Coello Coello, F. Luna, and E. Alba, "A comparative study of the effect of parameter scalability in multi-objective metaheuristics," in *Proc. IEEE Congr. Evol. Comput. (IEEE World Congress on Computational Intelligence)*, Hong Kong, 2008, pp. 1893–1900.

[159] J. J. Durillo, A. J. Nebro, C. A. Coello Coello, J. García-Nieto, F. Luna, and E. Alba, "A study of multiobjective metaheuristics when solving parameter scalable problems," *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, pp. 618–635, Aug. 2010.

[160] S. Liu, Q. Lin, K.-C. Wong, Q. Li, and K. C. Tan, "Evolutionary large-scale multiobjective optimization: Benchmarks and algorithms," *IEEE Trans. Evol. Comput.*, early access, Jul. 26, 2021, doi: 10.1109/TEVC.2021.3099487.

[161] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[162] X. Ma *et al.*, "A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables," *IEEE Trans. Evol. Comput.*, vol. 20, no. 2, pp. 275–298, Apr. 2016.

[163] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 97–112, Feb. 2018.

[164] A. E. I. Brownlee, J. A. Wright, M. He, T. Lee, and P. McMenemy, "A novel encoding for separable large-scale multi-objective problems and its application to the optimisation of housing stock improvements," *Appl. Soft Comput.*, vol. 96, Nov. 2020, Art. no. 106650.

[165] L. M. Antonio and C. A. Coello Coello, "Use of cooperative coevolution for solving large scale multiobjective optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, 2013, pp. 2758–2765.

[166] L. M. Antonio and C. A. Coello Coello, "Decomposition-based approach for solving large scale multi-objective problems," in *Parallel Problem Solving From Nature*. Cham, Switzerland: Springer, 2016, pp. 525–534.

[167] A. Song, Q. Yang, W.-N. Chen, and J. Zhang, "A random-based dynamic grouping strategy for large scale multi-objective optimization," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2016, pp. 468–475.

[168] B. Cao, J. Zhao, Y. Gu, Y. Ling, and X. Ma, "Applying graph-based differential grouping for multiobjective large-scale optimization," *Swarm Evol. Comput.*, vol. 53, Mar. 2020, Art. no. 100626.

[169] Q. Wang, L. Zhang, S. Wei, and B. Li, "Tensor decomposition-based alternate sub-population evolution for large-scale many-objective optimization," *Inf. Sci.*, vol. 569, pp. 376–399, Aug. 2021.

[170] L. Ma, M. Huang, S. Yang, R. Wang, and X. Wang, "An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization," *IEEE Trans. Cybern.*, early access, Jan. 21, 2021, doi: 10.1109/TCYB.2020.3041212.

[171] H. Chen, R. Cheng, J. Wen, H. Li, and J. Weng, "Solving large-scale many-objective optimization problems by covariance matrix adaptation evolution strategy with scalable small subpopulations," *Inf. Sci.*, vol. 509, pp. 457–469, Jan. 2020.

[172] A. Tiwari and R. Roy, "Variable dependence interaction and multi-objective optimisation," in *Proc. Genet. Evol. Comput. Conf.*, 2002, pp. 602–609.

[173] C. He *et al.*, "Accelerating large-scale multiobjective optimization via problem reformulation," *IEEE Trans. Evol. Comput.*, vol. 23, no. 6, pp. 949–961, Dec. 2019.

[174] H. Zille and S. Mostaghim, "Comparison study of large-scale optimisation techniques on the LSMOP benchmark functions," in *Proc. IEEE Symp. Ser. Comput. Intell.*, Honolulu, HI, USA, 2017, pp. 1–8.

[175] H. Zille, H. Ishibuchi, S. Mostaghim, and Y. Nojima, "Weighted optimization framework for large-scale multi-objective optimization," in *Proc. Genet. Evol. Comput. Conf.*, 2016, pp. 83–84.

[176] R. Liu, J. Liu, Y. Li, and J. Liu, "A random dynamic grouping based weight optimization framework for large-scale multi-objective optimization problems," *Swarm Evol. Comput.*, vol. 55, Jun. 2020, Art. no. 100684.

[177] Y. Tian, C. Lu, X. Zhang, K. C. Tan, and Y. Jin, "Solving large-scale multiobjective optimization problems with sparse optimal solutions via unsupervised neural networks," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3115–3128, Jun. 2021.

[178] Y. Tian, C. Lu, X. Zhang, F. Cheng, and Y. Jin, "A pattern mining-based evolutionary algorithm for large-scale sparse multiobjective optimization problems," *IEEE Trans. Cybern.*, early access, Dec. 30, 2020, doi: 10.1109/TCYB.2020.3041325.

[179] Y. Tian, X. Zhang, C. Wang, and Y. Jin, "An evolutionary algorithm for large-scale sparse multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 380–393, Apr. 2020.

[180] Y. Tian, R. Liu, X. Zhang, H. Ma, K. C. Tan, and Y. Jin, "A multi-population evolutionary algorithm for solving large-scale multimodal multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 25, no. 3, pp. 405–418, Jun. 2021.

[181] J.-H. Yi *et al.*, "Behavior of crossover operators in NSGA-III for large-scale optimization problems," *Inf. Sci.*, vol. 509, pp. 470–487, Jan. 2020.

[182] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.

[183] Y. Yin, Y. Zhao, H. Li, and X. Dong, "Multi-objective evolutionary clustering for large-scale dynamic community detection," *Inf. Sci.*, vol. 549, pp. 269–287, Mar. 2021.

[184] L. P. Cota *et al.*, "An adaptive multi-objective algorithm based on decomposition and large neighborhood search for a green machine scheduling problem," *Swarm Evol. Comput.*, vol. 51, Dec. 2019, Art. no. 100601.

[185] S. Qin, C. Sun, Y. Jin, Y. Tan, and J. Fieldsend, "Large-scale evolutionary multiobjective optimization assisted by directed sampling," *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 724–738, Aug. 2021.

[186] Y. Zhang, G.-G. Wang, K. Li, W.-C. Yeh, M. Jian, and J. Dong, "Enhancing MOEA/D with information feedback models for large-scale many-objective optimization," *Inf. Sci.*, vol. 522, pp. 1–16, Jun. 2020.

[187] W. Hong, K. Tang, A. Zhou, H. Ishibuchi, and X. Yao, "A scalable indicator-based evolutionary algorithm for large-scale multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 525–537, Jun. 2019.

[188] J. Xiao, T. Zhang, J. Du, and X. Zhang, "An evolutionary multiobjective route grouping-based heuristic algorithm for large-scale capacitated vehicle routing problems," *IEEE Trans. Cybern.*, vol. 51, no. 8, pp. 4173–4186, Aug. 2021.

[189] X. Zhang, K. Zhou, H. Pan, L. Zhang, X. Zeng, and Y. Jin, "A network reduction-based multiobjective evolutionary algorithm for community detection in large-scale complex networks," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 703–716, Feb. 2020.

[190] J. Zhang, L. Xing, G. Peng, F. Yao, and C. Chen, "A large-scale multiobjective satellite data transmission scheduling algorithm based on SVM + NSGA-II," *Swarm Evol. Comput.*, vol. 50, Nov. 2019, Art. no. 100560.

[191] S.-Y. Ho, L.-S. Shu, and J.-H. Chen, "Intelligent evolutionary algorithms for large parameter optimization problems," *IEEE Trans. Evol. Comput.*, vol. 8, no. 6, pp. 522–541, Dec. 2004.

[192] M. Gong, H. Li, E. Luo, J. Liu, and J. Liu, "A multiobjective cooperative coevolutionary algorithm for hyperspectral sparse unmixing," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 234–248, Apr. 2017.

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

[194] 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 Trans. Cybern.*, vol. 46, no. 4, pp. 1000–1013, Apr. 2016.

[195] A. Kumar, G. Wu, M. Z. Ali, R. Mallipeddi, P. N. Suganthan, and S. Das, "A test-suite of non-convex constrained optimization problems from the real-world and some baseline results," *Swarm Evol. Comput.*, vol. 56, Aug. 2020, Art. no. 100693.

[196] E. Mezura-Montes and C. A. Coello Coello, "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm Evol. Comput.*, vol. 1, no. 4, pp. 173–194, 2011.

[197] C. Peng and Q. Hui, "Comparison of differential grouping and random grouping methods on sCCPSO for large-scale constrained optimization," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2016, pp. 2057–2063.

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

[199] A. E. Aguilar-Justo and E. Mezura-Montes, "Towards an improvement of variable interaction identification for large-scale constrained problems," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2016, pp. 4167–4174.

[200] 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 *Proc. IEEE Int. Autumn Meeting Power Electron. Comput. (ROPEC)*, Ixtapa, Mexico, 2016, pp. 1–6.

[201] A. E. Aguilar-Justo and E. Mezura-Montes, "A local cooperative approach to solve large-scale constrained optimization problems," *Swarm Evol. Comput.*, vol. 51, Dec. 2019, Art. no. 100577.

[202] J. Blanchard, C. Beauthier, and T. Carletti, "A cooperative co-evolutionary algorithm for solving large-scale constrained problems with interaction detection," in *Proc. Genet. Evol. Comput. Conf.*, 2017, pp. 697–704.

[203] M. N. Omidvar, "IDG: A faster and more accurate differential grouping algorithm," Ph.D. dissertation, Dept. School Comput. Sci., Univ. Birmingham, Birmingham U.K., 2015.

[204] C. He, R. Cheng, Y. Tian, X. Zhang, K. C. Tan, and Y. Jin, "Paired offspring generation for constrained large-scale multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 25, no. 3, pp. 448–462, Jun. 2021.

[205] W. Chu, X. Gao, and S. Sorooshian, "Handling boundary constraints for particle swarm optimization in high-dimensional search space," *Inf. Sci.*, vol. 181, no. 20, pp. 4569–4581, 2011.

[206] O. A. Elhara, "Stochastic black-box optimization and benchmarking in large dimensions," Ph.D. dissertation, Dept. Laboratoire de Recherche en Informatique, Université Paris-Saclay, Gif-sur-Yvette, France, 2017.

[207] K. Tang *et al.*, "Benchmark functions for the CEC'2008 special session and competition on large scale global optimization," Dept. Nat. Inspired Comput. Appl. Lab., USTC, Hefei, China, Rep., 2007.

[208] 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," Dept. Nat. Inspired Comput. Appl. Lab., USTC, Hefei, China, Rep., 2009. [Online]. Available: http://nical.ustc.edu.cn/cec10ss.php

[209] 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 Univ., Melbourne, VIC, Australia, Rep., 2013, [Online]. Available: http://goanna.cs.rmit.edu.au/∼xiaodong/cec13-lsgo

[210] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimization problems," 2013, *arXiv:1308.4008*.

[211] P. Suganthan *et al.*, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Elect. Electron. Eng., Nanyang Technol. Univ., Singapore, Rep. 2005005, 2005. [Online]. Available: http://www.ntu.edu.sg/home/EPNSugan

[212] M. Lozano, D. Molina, and F. Herrera, "Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems," *Soft Comput.*, vol. 15, no. 11, pp. 2085–2087, 2011.

[213] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," INRIA, Le Chesnay-Rocquencourt, France, Rep. RR-6829, 2009.

[214] Y. Sun, M. Kirley, and S. K. Halgamuge, "Quantifying variable interactions in continuous optimization problems," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 249–264, Apr. 2017.

[215] E. Sayed, D. Essam, and R. Sarker, "Dependency identification technique for large scale optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, Brisbane, QLD, Australia, 2012, pp. 1–8.

[216] S. K. Goh, K. C. Tan, A. Al-Mamun, and H. A. Abbass, "Evolutionary big optimization (BigOpt) of signals," in *Proc. IEEE Congr. Evol. Comput.*, Sendai, Japan, 2015, pp. 3332–3339.

[217] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "Test problems for large-scale multiobjective and many-objective optimization," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4108–4121, Dec. 2017.

[218] M. N. Omidvar, X. Li, and K. Tang, "Designing benchmark problems for large-scale continuous optimization," *Inf. Sci.*, vol. 316, pp. 419–436, Sep. 2015.

[219] E. D. Dolan, J. J. More, and T. S. Munson, "Benchmarking optimization software with COPS 3.0," Dept. Math. Comput. Sci. Division, Argonne Nat. Lab., Argonne, IL, USA, Rep. ANL/MCS-TM-273, 2004.

[220] W. Luo, B. Yang, C. Bu, and X. Lin, "A hybrid particle swarm optimization for high-dimensional dynamic optimization," in *Proc. Asia–Pac. Conf. Simul. Evol. Learn.*, 2017, pp. 981–993.

[221] L.-Y. Tseng and C. Chen, "Multiple trajectory search for large scale global optimization," in *Proc. IEEE Congr. Evol. Comput.*, Hong Kong, 2008, pp. 3052–3059.

[222] Y. Wang and B. Li, "A restart univariate estimation of distribution algorithm: Sampling under mixed Gaussian and lévy probability distribution," in *Proc. IEEE Congr. Evol. Comput.*, Hong Kong, 2008, pp. 3917–3924.

[223] Y. Wang and B. Li, "Two-stage based ensemble optimization for large-scale global optimization," in *Proc. IEEE Congr. Evol. Comput.*, Barcelona, Spain, 2010, pp. 1–8.

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

[225] A. LaTorre, S. Muelas, and J.-M. Peña, "Large scale global optimization: Experimental results with MOS-based hybrid algorithms," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, 2013, pp. 2742–2749.

[226] D. Molina and F. Herrera, "Iterative hybridization of DE with local search for the CEC'2015 special session on large scale global optimization," in *Proc. IEEE Congr. Evol Comput.*, Sendai, Japan, 2015, pp. 1974–1978.

[227] D. Molina, A. LaTorre, and F. Herrera, "SHADE with iterative local search for large-scale global optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Rio de Janeiro, Brazil, Jul. 2018, pp. 1–8.

[228] A. A. Hadi, A. W. Mohamed, and K. M. Jambi, "LSHADE-SPA memetic framework for solving large-scale optimization problems," *Complex Intell. Syst.*, vol. 5, no. 1, pp. 25–40, 2019.

[229] A. Bolufé-Röhler, S. Chen, and D. Tamayo-Vera, "An analysis of minimum population search on large scale global optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Wellington, New Zealand, 2019, pp. 1228–1235.

[230] Y. Wang *et al.*, "Two-stage based ensemble optimization framework for large-scale global optimization," *Eur. J. Oper. Res.*, vol. 228, no. 2, pp. 308–320, 2013.

[231] D. Molina, A. LaTorre, and F. Herrera, "An insight into bio-inspired and evolutionary algorithms for global optimization: Review, analysis, and lessons learnt over a decade of competitions," *Cogn. Comput.*, vol. 10, no. 4, pp. 517–544, 2018.

[232] D. M. Cabrera, "Evolutionary algorithms for large-scale global optimisation: A snapshot, trends and challenges," *Progr. Artif. Intell.*, vol. 5, no. 2, pp. 85–89, 2016.

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

[234] K. Sörensen, "Metaheuristics—The metaphor exposed," *Int. Trans. Oper. Res.*, vol. 22, no. 1, pp. 3–18, 2015.

[235] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," *Nat. Mach. Intell.*, vol. 1, no. 1, pp. 24–35, 2019.

[236] 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," 2018, *arXiv:1712.06567*.

[237] S. Fujino, T. Hatanaka, N. Mori, and K. Matsumoto, "The evolutionary deep learning based on deep convolutional neural network for the anime storyboard recognition," in *Proc. Int. Symp. Distrib. Comput. Artif. Intell.*, 2017, pp. 278–285.

[238] G. Morse and K. O. Stanley, "Simple evolutionary optimization can rival stochastic gradient descent in neural networks," in *Proc. Genet. Evol. Comput. Conf.*, 2016, pp. 477–484.

[239] E. Real *et al.*, "Large-scale evolution of image classifiers," in *Proc. 34th Int. Conf. Mach. Learn. Vol. 70*, 2017, pp. 2902–2911.

[240] A. Yaman, D. C. Mocanu, G. Iacca, G. Fletcher, and M. Pechenizkiy, "Limited evaluation cooperative co-evolutionary differential evolution for large-scale neuroevolution," in *Proc. Genet. Evol. Comput. Conf.*, 2018, pp. 569–576.

[241] X. Cui, W. Zhang, Z. Tüske, and M. Picheny, "Evolutionary stochastic gradient descent for optimization of deep neural networks," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2018, pp. 6048–6058.

[242] M. Jaderberg *et al.*, "Population based training of neural networks," 2017, *arXiv:1711.09846*.

[243] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, pp. 1–21, Mar. 2019.

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

[245] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *Eur. J. Oper. Res.*, vol. 290, no. 2, pp. 405–421, 2021.

[246] A. Lodi and G. Zarpellon, "On learning and branching: A survey," *TOP*, vol. 25, no. 2, pp. 207–236, 2017.

[247] Y. Sun, X. Li, and A. Ernst, "Using statistical measures and machine learning for graph reduction to solve maximum weight clique problems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 5, pp. 1746–1760, May 2021.

[248] Y. Sun, A. Ernst, X. Li, and J. Weiner, "Generalization of machine learning for problem reduction: A case study on travelling salesman problems," *OR Spectr.*, vol. 43, no. 3, pp. 607–633, 2021.

[249] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*. Belmont, MA, USA: Athena Sci., 1997.

[250] M. Kruber, M. E. Lübbecke, and A. Parmentier, "Learning when to use a decomposition," in *Integration of AI and OR Techniques in Constraint Programming* (Lecture Notes in Computer Science), D. Salvagnin and M. Lombardi, Eds. Cham, Switzerland: Springer Int., 2017, pp. 202–210.

[251] C. Blum and G. R. Raidl, *Hybrid Metaheuristics: Powerful Tools for Optimization*. Cham, Switzerland: Springer, 2016.

[252] A. Kenny, X. Li, A. T. Ernst, and D. Thiruvady, "Towards solving large-scale precedence constrained production scheduling problems in mining," in *Proc. Genet. Evol. Comput. Conf.*, 2017, pp. 1137–1144.

[253] A. Kenny, X. Li, and A. T. Ernst, "A merge search algorithm and its application to the constrained pit problem in mining," in *Proc. Genet. Evol. Comput. Conf.*, 2018, pp. 316–323.

[254] A. Kenny, X. Li, A. T. Ernst, and Y. Sun, "An improved merge search algorithm for the constrained pit problem in open-pit mining," in *Proc. Genet. Evol. Comput. Conf.*, 2019, pp. 294–302.

[255] S. Elsayed, R. Sarker, D. Essam, and C. A. Coello Coello, "Evolutionary approach for large-Scale mine scheduling," *Inf. Sci.*, vol. 523, pp. 77–90, Jun. 2020.

[256] C. He, R. Cheng, C. Zhang, Y. Tian, Q. Chen, and X. Yao, "Evolutionary large-scale multiobjective optimization for ratio error estimation of voltage transformers," *IEEE Trans. Evol. Comput.*, vol. 24, no. 5, pp. 868–881, Oct. 2020.

Mohammad Nabi Omidvar (Senior Member, IEEE) received the first bachelor's degree (First Class Hons.) in computer science, the second bachelor's degree in applied mathematics, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 2010, 2014, and 2016, respectively.

He is currently an Assistant Professor of Artificial Intelligence in Financial Services with the Leeds University Business School and School of Computing, University of Leeds, Leeds, U.K. Prior to that, he was a Research Fellow with the School of Computer Science, University of Birmingham, Birmingham, U.K. His current research interests include large-scale global optimization, high-dimensional machine learning, and AI for financial services.

Dr. Omidvar is the winner of IEEE CEC Large-Scale Global Optimization Competition in 2019. He was a recipient of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award for his research on large-scale global optimization in 2017, the Australian Postgraduate Award in 2010, and the Best Computer Science Honours Thesis Award from the School of Computer Science and IT, RMIT University. He was the Chair of IEEE Computational Intelligence Taskforce on Large-Scale Global Optimization.

Xiaodong Li (Fellow, IEEE) received the B.Sc. degree from Xidian University, Xi'an, China, in 1988, and the Ph.D. degree in information science from the University of Otago, Dunedin, New Zealand, in 1998.

He is a Professor with the School of Computing Technologies, RMIT University, Melbourne, VIC, Australia. His research interests include machine learning, evolutionary computation, neural networks, data analytics, multiobjective optimization, multimodal optimization, and swarm intelligence.

Prof. Li was a recipient of the 2013 ACM SIGEVO Impact Award and the 2017 IEEE CIS IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award. He serves as an Associate Editor for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and *Swarm Intelligence* (Springer). He is the Former Vice-Chair of the IEEE Task Force on Multimodal Optimization and the Former Chair of the IEEE CIS Task Force on Large Scale Global Optimization.

Xin Yao (Fellow, IEEE) received the B.Sc. degree from the University of Science and Technology of China (USTC), Hefei, China, in 1982, the M.Sc. degree from the North China Institute of Computing Technologies, Beijing, China, in 1985, and the Ph.D. degree from USTC in 1990.

He is the Chair Professor of Computer Science with the Southern University of Science and Technology, Shenzhen, China, and a Part-Time Professor of Computer Science with the University of Birmingham, Birmingham, U.K. His major research interests include evolutionary computation, ensemble learning, and their applications to software engineering.

Dr. Yao received the Prestigious Royal Society Wolfson Research Merit Award in 2012, the IEEE CIS Evolutionary Computation Pioneer Award in 2013, and the 2020 IEEE Frank Rosenblatt Award. His work won the 2001 IEEE Donald G. Fink Prize Paper Award; 2010, 2016, and 2017 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Awards; 2011 IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award; and many other best paper awards at conferences. He served as the President of IEEE Computational Intelligence Society (CIS) from 2014 to 2015 and the Editor-in-Chief of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2003 to 2008. He was a Distinguished Lecturer of IEEE CIS.