Research Article

A Novel Memetic Algorithm Based on Multiparent Evolution and Adaptive Local Search for Large-Scale Global Optimization

Wenfen Zhang1,2 and Yulin Lan1,2

1School of Computer and Artificial Intelligence, Xiangnan University, Chenzhou, China
2Hunan Engineering Research Center of Advanced Embedded Computing and Intelligent Medical Systems, Chenzhou, China

Correspondence should be addressed to Yulin Lan; lany0000@126.com

Received 26 August 2021; Revised 22 December 2021; Accepted 24 February 2022; Published 24 March 2022

Academic Editor: Paulo Moura Oliveira

Copyright © 2022 Wenfen Zhang and Yulin Lan. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In many fields, including management, computer, and communication, Large-Scale Global Optimization (LSGO) plays a critical role. It has been applied to various applications and domains. At the same time, it is one of the most challenging optimization problems. This paper proposes a novel memetic algorithm (called MPCE & SSALS) based on multiparent evolution and adaptive local search to address the LSGO problems. In MPCE & SSALS, a multiparent crossover operation is used for global exploration, while a step-size adaptive local search is utilized for local exploitation. A new offspring is generated by recombining four parents. In the early stage of the algorithm execution, global search and local search are performed alternately, and the population size gradually decreases to 1. In the later stage, only local searches are performed for the last individual. Experiments were conducted on 15 benchmark functions of the CEC’2013 benchmark suite for LSGO. The results were compared with four state-of-the-art algorithms, demonstrating that the proposed MPCE & SSALS algorithm is more effective.

1. Introduction

Optimization problems widely exist in many fields such as engineering design, economic management, production scheduling [1–4], wireless communication, and computer science. Some of these problems have many decision variables that create Large-Scale Global Optimization (LSGO) problems. Without loss of generality, a large-scale global optimization can be formulated as a minimization problem as given in

$$\min_{x \in \text{Domain}} f(x), \text{Domain} \subseteq \mathbb{R}^{D}, \quad (1)$$

where $D \geq 1000$ is the dimension size.

Large-scale global optimization is one of the most challenging optimization problems where the search space grows exponentially with increasing the problem dimensionality. Such a massive increase in problem dimensions usually changes the search properties. Consequently, the small-scale unimodal function may change to a multimodal function when the number of dimensions increases [5]. Therefore, researchers have proposed many improved algorithms based on the existing classic algorithms. For example, in [6,7], the Particle Swarm Optimization (PSO) algorithm was improved using subswarms to maintain diversity. Also, in [8], an improved PSO algorithm containing two types of learning strategies was provided. Multiple Offspring Sampling (MOS) [9] is a hybrid algorithm that combines a Genetic Algorithm (GA) and two local searches. MLSHADE-SPA [5] also is a hybrid algorithm that is based on three Differential Evolution (DE) strategies and a modified Multiple Trajectory Search (MTS) [10] algorithm. IMLSHADE-SPA [11] is an improved MLSHADE-SPA with a novel local search method. However, SHADE-ILS [12] is an enhanced version of the SHADE algorithm. It combines two different local search methods and uses a restart mechanism. Algorithms in [13–16] are modified algorithms based on Cooperative Coevolution (CC) and Differential Evolution. CBCC-RDG3 [17] is also a modified version of the CC algorithm that modifies the recursive differential
grouping method to reduce the overlapping problems. TPHA [18] and DECC-RAG1.1 [19] are two-phase hybrid algorithms that use the CC framework.

To encourage the research on LSGO, IEEE Congress on Evolutionary Computation (IEEE CEC) organizes LSGO algorithm competitions yearly or biennial. Since 2013, it has been performed on the CEC’2013 LSGO benchmark suite [20]. MOS was the winner in the years 2013–2018. Moreover, MOS and the other 11 excellent algorithms that did not join the competitions were compared in [21]. Again, the MOS algorithm overperformed all the other algorithms. However, SHADE-ILS and MLSHADE-SPA were announced in the 2018 competition that they are performing better than MOS. Although CC-RDG3 [17] was announced as the winner of the 2019 competitions, it was not up to the level of the previous winner, SHADE-ILS. It seems that LSGO is still a quite hard nut to crack [21].

The algorithms for LSGO problems can be roughly classified into three categories: standard evolutionary algorithms, CC-based evolutionary algorithms, and memetic algorithms [22]. The memetic algorithm (MA) [23] is a combination of global search and local search. Due to the exploration ability of global search and the exploitation ability of local search, MA performs well in LSGO problems. As mentioned above, the CEC competition award algorithms (e.g., SHADE-ILS and MLSHADE-SPA) and improved algorithm IMLSHADE-SPA based on MLSHADE-SPA are all MAs. In MLSHADE-SPA, IMLSHADE-SPA, and SHADE-ILS, global search and local search have the same position, and the two search methods generate the same number of candidate solutions. However, since the dimension exceeds 1000 for LSGO problems, it is necessary to discuss the case that the numbers of local search and global search are different. Furthermore, at each iteration of the algorithms, the local search is only used to improve the current best solution, and other members cannot be improved, which may miss potential excellent individuals. Besides, the three algorithms all used Differential Evolution and adopted a variety of improvement strategies, which makes the algorithm more complicated. In this case, it is necessary to examine new algorithms and new ideas. This paper proposes a novel memetic algorithm (called MPCE & SSALS) based on multiparent evolution and local search. A multiparent crossover operator is used for global exploration. Furthermore, a step-size adaptive local search algorithm, which is improved from the MTS algorithm, is proposed for local exploitation. The proposed algorithm is inspired by the Simplified Group Search Optimizer (SGSO) [24] algorithm to generate parent vectors and adopt a population size reduction strategy. There are three main differences between the proposed algorithm and the above memetic algorithms: (1) The proposed algorithm performs much more local searches than global searches. (2) The proposed algorithm performs the local search for every individual. (3) The proposed algorithm uses a new and simpler global search method.

In the following sections, the details of the MPCE & SSALS algorithm will be explained. The algorithm is also compared with four state-of-the-art algorithms, namely, SHADE-ILS, MLSHADE-SPA, CBCC-RDG3, and IMLSHADE-SPA.

The main contributions and novelty of this paper can be summarized as follows:

(i) Proposing a novel memetic algorithm for the LSGO problem

(ii) Using multiparent crossover and SGSO to solve LSGO problem

(iii) Proposing an improved local search algorithm that can be an effective option for LSGO

(iv) Demonstrating that local search-dominated hybrid algorithm can effectively solve the LSGO problems.

The rest of this paper is organized as follows. In Section 2, the most related work to this paper proposed approach is discussed. Section 3 presents the details of the proposed algorithm. Section 4 explains the numerical experiments of MPCE & SSALS that are carried out using the CEC’2013 benchmark suite and the performance of MPCE & SSALS compared to four algorithms. Finally, conclusions are made, and further research is discussed in Section 5.

2. Related Works

This section is devoted to presenting the related work needed for understanding the MPCE & SSALS algorithm. Memetic algorithms, multiparent crossover, simplified group search optimizer, and MTS are described.

2.1. Memetic Algorithms. Moscato [23] first proposed the concept of memetic algorithm in 1989. The memetic algorithm is a combination of population-based global search and individual-based heuristic local search. It suggests an algorithm framework. In this framework, different search strategies are used to construct different memetic algorithms. For example, Genetic algorithms, Differential Evolution, Particle Swarm Optimization, and many others can be used for global search strategy. Hill Climbing, Simulated Annealing, Tabu Search, and others can be used for local search strategy. Memetic algorithms have embraced many forms, employing a wide variety of combinations of population-based heuristics and individual improvement heuristics [25], such as [26–33]. Some of these algorithms based on GA and Tabu Search were studied in [26,27]. The memetic model of PSO and local search was introduced in [28–30]. A Memetic Artificial Bee Colony Algorithm is also reported in [31]. The combination of a backbone-based crossover operator and a multi-neighborhood simulated annealing procedure was discussed in [32]. In [33], adaptive memetic computing with a GA, DE, and Estimation of Distribution Algorithm synergy was elaborated. It can automatically activate one of the three algorithms to generate offspring.

Memetic algorithms combine the exploration capability of population-based global search with the rapid exploitation capability of local search. The hybridization between global and local search algorithms has been experimentally proven to provide better search performance [33]. Many examples have proved the effectiveness of this strategy,
including many difficult problems, such as multimodal optimization [33,34], large-scale global optimization [5–19], combinatorial optimization [32,35], single-objective optimization [36,37], and multiobjective optimization [38,39]. For large-scale global optimization problems, most of the best-performing algorithms were hybrid algorithms combining global and local search (e.g., SHADE-ILS, MLSHADE-SPA, and MOS).

SHADE with an iterative Local Search (SHADE-ILS) is a hybrid algorithm that combines a modern DE algorithm, Success-History-based Adaptive DE (SHADE [40]), with two local search methods. In each iteration, the SHADE is applied to evolve the population of candidate solutions. One of the two local search methods is chosen to improve the current best solution found by SHADE. The local search method’s selection is according to the improvement obtained by each of them in the previous phase. A restart mechanism has been incorporated into the algorithm to explore new search space regions when the search gets stagnated.

MLSHADE-SPA is a memetic framework that includes three DE algorithms for global exploration and a modified version of MTS (MMTS) for local exploitation. The three DE algorithms are success history-based differential evolution with linear population size reduction and semiparameter adaptation (LSHADE-SPA), enhanced adaptive differential evolution (EADE) [41], and differential evolution with novel mutation and adaptive crossover strategies (ANDE) [42]. The framework also uses the divide-and-conquer method, which randomly divides the dimensions into groups and solves each group separately.

An improved MLSHADE-SPA (IMLSHADE-SPA) framework was proposed in [11], which replaced the local search method (MMTS) with a new local search method and achieved higher performance.

Multiple Offspring Sampling (MOS) [43] is a framework used to combine different metaheuristic algorithms. The participation ratio for each algorithm is adjusted dynamically according to a given strategy. Due to the different algorithms and strategies to be selected, different MOS versions were proposed in [9,44,45]. Our paper focuses on the MOS [9], the winner in the CEC’2013 competition. In MOS [9], three algorithms are combined: GA, Solis and Wets algorithm [46], and MTS-LS1-Reduced algorithm. These algorithms are executed in sequence, one after the other. The number of candidate solutions to be generated by each algorithm is adjusted dynamically according to the average fitness increment of the newly created individuals [9].

### 2.2. Multiparent Crossover

Evolutionary algorithms (EAs) have been successfully applied to solve many optimization problems. EAs simulate the evolution process of nature. There are three basic operators in EAs: crossover (or recombination), mutation, and selection. The classic crossover operator recombines two parents and generates new offspring. The recombination mechanism determines what parts of each parent are inherited by the child and how this is done. Various crossover operators were proposed for different problems that fit one of the multiple representations for a chromosome [47]. These crossover operators can be categorized into two categories: exchange-based or calculation-based. The first type of operator is generally proposed for binary coding, but it is also suitable for real coding. Some of the crossover operators’ examples are One-point Crossover, Two-point Crossover, Uniform Crossover, and so on. For instance, Uniform Crossover randomly determines whether the child’s ith gene is selected from father 1 or father 2. With these crossover mechanisms, each gene in the offspring is copied from one of the parents. The new offspring’s chromosome characteristics are directly inherited from their parents without any changes.

The second category of the crossover operators is generally used for real coding, such as Average Crossover, Parent Centric Crossover, Heuristic Crossover, Simulated Binary Crossover, and so on. In these operators, the value of each offspring’s gene is calculated numerically by the parent’s genes. For example, Average Crossover generates the ith gene of the child by averaging alleles from both parents.

The first category is more in line with the original concept of gene recombination. In some algorithms, such as the DE algorithm, the second category crossover operator is used as a mutation rather than a crossover operator. It can produce new genes that are different from their parents. This paper tends to define the second type of operators as a hybrid of crossover and mutation operations.

Multiparent crossover extends the two-parent crossover operators to recombine more than two parents for generating new offspring. Many multiparent crossovers have been successfully applied to solve various optimization problems and found to be better than traditional crossovers, such as scanning crossover and diagonal crossover [48,49], multiparent simplex crossover [50], multiparent sequential constructive crossover [47], and a novel multiparent order crossover [51].

#### 2.3. Simplified Group Search Optimizer

Group Search Optimizer (GSO) is a swarm intelligence algorithm with superior performance for multimodal problems [52].

GSO is inspired by animal searching behaviors and group living theory [52]. It includes three types of members: producer, scrounger, and ranger. During each iteration, the individual with the best fitness value in the group, as a producer, will stop and scan the environment to find resources. The scrounger takes a random walk towards the producer to join the resources. A small number of rangers make a random move to avoid entrapment in local minima.

The Simplified Group Search Optimizer (SGSO) [24] is an improved GSO version. It is more efficient and simpler than the original version. It also shows excellent search performance for large-scale optimization problems. In SGSO, the producer abandons environmental scanning. The scrounger adopts an improved join strategy, which moves towards the best member and other excellent members. The rangers use a simple search method and the ranger’s percentage decreases. The SGSO is described as follows:

1. In a D-dimensional search space, the ith member at the kth iteration has a current position, \( x_{ik} \in \mathbb{R}^D \).
(2) Group members are sorted by fitness value in ascending order. The best member $x_{\text{best},k}$ as the producer, does not move in this iteration.

(3) Randomly select 87% of the group members except the producer to perform scrounging. The scroungers move to a new position according to

$$x_{i,k+1} = x_{i,k} + r_1 (x_{\text{best},k} - x_{i,k}) + r_2 (x_{\text{m-best},k} - x_{i,k}),$$

where $r_1$ and $r_2$ are uniform random $D$-dimensional vectors in the range $(0, 1)$ and $x_{\text{m-best},k}$ is a member randomly chosen from the top 4 in the group (except $x_{\text{best},k}$).

(4) The remaining members are rangers, who take a random step according to

$$x_{i,k+1} = x_{i,k} + r_i \cdot f \cdot \text{step},$$

where $r_i$ is a standard normal distribution $D$-dimensional vector, step is a constant, representing the basic step size, and $f$ is a $D$-dimensional Boolean random vector indicating which dimensions will change. The probability of change is set to be $1.2/D$ as given in [24].

(5) $f$ is calculated by

$$f_j = \begin{cases} 1, & \text{if } \text{rand}(1) \leq \frac{1.2}{D} \text{ or } j = j_{\text{rand}}, \\ 0, & \text{otherwise}, \end{cases}$$

where $j \in \{1, 2, \ldots, D\}$, rand(1) is a function that produces a uniform random number in the range $(0, 1)$, and $j_{\text{rand}}$ is a randomly chosen index $\in \{1, 2, \ldots, D\}$, which ensures that at least one component in $f$ is set to 1.

2.4. Multiple Trajectory Search. Multiple trajectory search (MTS) was presented for the large-scale global optimization problem in [10]. It provides three local search methods, where MTS-LS1 is the first and most important one.

MTS-LS1 does search from the first to the last dimension successively. Each dimension is subtracted from the search range (SR) value to see whether the objective function value is improved or not. If it is improved, MTS-LS1 proceeds to search in the next dimension. If it is not improved, the solution is restored, and this dimension is added by $0.5 \times \text{SR}$, aiming to see, again, if its value is improved or not. If it is not improved, the solution is restored. Afterward, MTS-LS1 continues to search in the next dimension. SR is initialized to $0.5 \times \text{SR}$. If all dimensions are not improved, SR will be cut to half. When SR reaches $1E-15$, its value will be reset to $0.4 \times \text{(Upper_Bound} - \text{Lower_Bound)}$. MTS-LS1 and its improved versions are used in many algorithms [5,9,12], including the algorithm proposed in this paper.

3. Proposed Algorithm

In this section, multiparent Crossover Evolution and Step-Size Adaptive Local Search algorithm will be described. Besides, the proposed hybrid algorithm that combines both of them will be introduced.

3.1. Multiparent Crossover Evolution (MPCE). In MPCE, the population is composed of $D$-dimensional vectors. The number of vectors is called population size, denoted as NP. The initial population is generated with uniformly distributed random numbers. Each member of the population can produce the next generation through mutation and multi-parent crossover operation. The $i$th member of the $G$th generation is denoted as $x_{i,G}$.

The main characteristics of MPCE are as follows:

(1) The mutation formula is modified from (2) of the SGSO algorithm. The mutant vector is generated according to

$$v_{i,G} = x_{i,G} + r_1 (x_{\text{best},G} - x_{i,G}) + r_2 (x_{p\text{-best},G} - x_{i,G}),$$

where $x_{\text{best},G}$ is the best vector in the $G$th generation, $p\text{-best}$ is the index of a vector which is randomly chosen from the ranked top 10% vectors in the $G$th generation (except $x_{\text{best},G}$), and $r_1$ and $r_2$ are uniform random vectors in the range $(0, 1)$.

(2) MPCE uses a four-parent crossover operation to produce the next generation. The four parents are $x_i$, $v_j$, and two excellent individuals randomly selected from the population. The crossover operation could be computed using

$$x_{ji,G+1} = \begin{cases} v_{ji,G}, & \text{if } r_{\text{rand}(ji)} \leq \text{CP1}, \\ x_{ja(i),G}, & \text{if } \text{CP1} < r_{\text{rand}(ji)} \leq \text{CP1} + \text{CP2}, \\ x_{jb(i),G}, & \text{if } \text{CP1} + \text{CP2} < r_{\text{rand}(ji)} \leq \text{CP1} + \text{CP2} + \text{CP3}, \\ x_{ji,G}, & \text{otherwise}, \end{cases}$$

where $j \in \{1, 2, \ldots, D\}$, $a(i)$ and $b(i) \in \{1, 2, \ldots, \text{NP}\}$ are the index of vectors randomly chosen from ranked top 50% vectors in the $G$th generation, CP1, CP2, CP3 $\in (0,1)$ are the crossover probability constant of $v_j$, $x_{a(i)}$, $x_{b(i)}$, respectively, $r_{\text{rand}(ji)}$ is a uniform random number in the range $(0, 1)$. 


The parameters CP1, CP2, and CP3 are determined through experiments and set to 0.3, 0.29, and 0.29, respectively.

(3) The population size decreases during the optimization process. With the ongoing iteration, the vectors in the population tend to be gradually assimilated, where the larger NP is less helpful to improve the search performance. Many algorithms apply the population size linear decrease strategies, such as LSHADE algorithm [53]. Besides, for the MPCE & SSALS algorithm, reducing the population size is conducive to deeper local search. In the beginning, the global search and the local search are performed alternately. NP is reduced by 1, and the worst individual in the population is dismissed every several iterations. When NP is reduced to 4, MPCE global search ends and then only the local search is executed to improve the current best solution.

3.2. Step-Size Adaptive Local Search (SSALS). The basic idea of SSALS derives from MTS-LS1 that is the first local search strategy in the Multiple Trajectory Search (MTS) algorithm. These algorithms are designed for single individuals and can also be used for multiple individuals when combined with other algorithms.

Each dimension of the SSALS algorithm has its basic step size, stored in the vector \( s \). In each iteration, SSALS randomly selects one or more dimensions, multiplies the step size of each dimension by a random number, and adds the product to each dimension. If the new solution is better than the original one, these selected dimensions’ step size is multiplied by 2. Otherwise, the solution is restored, and each step size is multiplied by \(-0.5\). The step size is initialized using \( 0.5 \cdot (\text{Upper\_Bound} - \text{Lower\_Bound}) \). The variable \( \text{minbs} \) represents the minimum step size, an adaptive value that is recalculated in each iteration. If the step size’s absolute value reaches \( \text{minbs} \), it will be restored using the initial value.

In the case of multiple individuals, the SSALS algorithm key steps are described as follows.

1. Choose dimensions to be searched according to

   \[
   f_{ji,G} = \begin{cases} 
   1, & \text{if } \text{rand}(1) < \frac{0.5}{\text{D}} + \left(\frac{0.1 \cdot \text{D}}{\text{iteration}}\right)^2 \text{ or } j = j_{\text{rand}(i)}, \\
   0, & \text{otherwise,}
   \end{cases}
   \]

   where \( j_{\text{rand}(i)} \in \{1, 2, \ldots, \text{D}\} \) is a randomly chosen index to ensure that \( x_i \) has at least one dimension to participate in the search.

According to (7), the number of dimensions to be searched will rapidly decrease in the iterative process and finally keep at 1.5 per vector on average. This value makes the algorithm has a bit of global search ability in the early stage of the optimization process.

2. Generate the new solution according to

   \[
   x_{i,G+1} = x_{i,G} + s_{i,G} \cdot \text{rand}((\text{D}, 1)) \cdot f_{i,G},
   \]

   where \( s_{i,G} \) is a vector, representing the basic step size of the \( i \)th individual in the \( G \)th generation.

3. Calculate the variate \( \text{minbs} \). SSALS defines a \( \text{Dx5} \) matrix \( H \), which is used to store each dimension’s last five effective step sizes. The effective step size is defined as

   \[
   e_{ji,G+1} = \begin{cases} 
   |x_{ji,G+1} - x_{ji,G}|, & \text{if } x_{ji,G+1} \text{ is better than } x_{ji,G}, \\
   \text{null}, & \text{otherwise,}
   \end{cases}
   \]

   If more than one vector is improved in an iteration, and some of these vectors’ same dimension is changed, save the average effective step size of this dimension into \( H \).

   The formula for calculating \( \text{minbs} \) is given in

   \[
   \text{minbs} = \text{min}(0.1, \text{min} (\text{mean}(H, 2))).
   \]

4. Update the basic step size according to

   \[
   s_{ji,G+1} = \begin{cases} 
   s_{ji,G} \times (-0.5), & \text{if } x_{ji,G} \text{ is better than } x_{ji,G+1}, \text{ and } f_{ji,G} = 1, \\
   s_{ji,G} \times 2, & \text{if } x_{ji,G} \text{ is not better than } x_{ji,G+1}, \text{ and } f_{ji,G} = 1, \\
   s_{ji,G}, & \text{otherwise.}
   \end{cases}
   \]
3.3. Hybrid Algorithm: MPCE and SSALS. MPCE & SSALS is a memetic algorithm based on MPCE and SSALS, presented in Algorithm 1. The hybrid strategy is to perform one global search iteration after a certain number of local search iterations. The constant $I_{GS}$ indicates the interval of global search. Also, the constant $I_{NPD}$ refers to the decreased interval of NP. In other words, NP is subtracted by 1 for each $I_{NPD}$ iteration. When NP is reduced to 4, the global search ends, and only the local search is performed to improve the current best solution. The parameters $I_{GS}$ and $I_{NPD}$ are determined by experiments in which they are set to 40 and 100, respectively.

MPCE & SSALS performs a boundary check on each new individual generated, and all values outside the boundary will be reset to random numbers within this boundary. The pseudocodes of the proposed algorithms are given in Algorithm 1.

4. Experimentation

A set of 15 benchmark functions proposed in the CEC 2013 special session on large-scale global optimization was used to study the MPCE & SSALS performance. These functions are divided into four categories according to the degree of separability. $f_1$–$f_3$ belong to fully separable functions, $f_4$–$f_{11}$
| Milestone | Category | $f_1$  | $f_2$  | $f_3$  | $f_4$  | $f_5$  | $f_6$  | $f_7$  | $f_8$  | $f_9$  | $f_{10}$ | $f_{11}$ | $f_{12}$ | $f_{13}$ | $f_{14}$ | $f_{15}$ |
|-----------|----------|--------|--------|--------|--------|--------|--------|--------|--------|--------|----------|----------|----------|----------|----------|----------|
| $1.2E+05$ | Mean     | $6.27E+09$ | $1.73E+04$ | $1.94E+01$ | $5.47E+11$ | $1.00E+07$ | $2.95E+05$ | $7.26E+09$ | $1.01E+16$ | $7.38E+08$ | $1.61E+06$ | $1.31E+12$ | $2.30E+11$ | $2.96E+11$ | $2.74E+12$ | $2.62E+08$ |
| $6.0E+05$ | Mean     | $7.90E+01$ | $7.95E+02$ | $1.07E+00$ | $3.99E+10$ | $1.18E+06$ | $7.46E+02$ | $2.32E+08$ | $2.24E+15$ | $9.02E+07$ | $5.00E+04$ | $4.11E+10$ | $2.83E+03$ | $4.65E+09$ | $7.84E+10$ | $7.56E+06$ |
|          | Best     | $0.00E+00$ | $1.57E+01$ | $3.34E+01$ | $6.70E+13$ | $6.89E+07$ | $7.40E+03$ | $2.45E+05$ | $1.48E+02$ | $1.44E+05$ | $4.95E+06$ | $5.09E+05$ | $1.28E+02$ | $1.01E+06$ | $1.36E+05$ | $1.36E+05$ |
|          | Median   | $3.94E+28$ | $9.95E+01$ | $3.59E+13$ | $3.29E+09$ | $1.22E+06$ | $7.20E+02$ | $4.10E+03$ | $1.86E+14$ | $8.94E+07$ | $2.37E+04$ | $2.63E+06$ | $1.31E+01$ | $1.36E+06$ | $5.41E+06$ | $5.86E+05$ |
| $3.0E+06$ | Worst    | $1.51E+26$ | $3.98E+00$ | $3.77E+13$ | $5.55E+09$ | $1.46E+06$ | $1.40E+03$ | $7.27E+03$ | $3.00E+14$ | $1.15E+08$ | $1.43E+05$ | $1.51E+07$ | $2.24E+02$ | $2.14E+07$ | $6.27E+06$ | $6.49E+05$ |
|          | Mean     | $1.53E+27$ | $1.35E+00$ | $3.59E+13$ | $3.56E+09$ | $1.18E+06$ | $7.46E+02$ | $4.39E+03$ | $1.84E+14$ | $9.02E+07$ | $4.98E+04$ | $3.32E+06$ | $5.85E+01$ | $3.56E+06$ | $5.46E+06$ | $5.80E+05$ |
|          | Std      | $3.24E+27$ | $1.34E+00$ | $9.94E+15$ | $9.66E+08$ | $1.38E+05$ | $2.81E+02$ | $1.40E+03$ | $4.97E+13$ | $1.21E+07$ | $4.81E+04$ | $3.49E+06$ | $6.71E+01$ | $5.66E+06$ | $3.42E+05$ | $3.59E+04$ |

Table 1: MPCE & SSALS statistical result on the CEC'2013 LSGO functions, $D=1000$, $FES = 3.0E+06$. 

Comptational Intelligence and Neuroscience 7
are partially separable functions, f12–f14 belong to overlapping functions, and f15 is classified as fully nonseparable functions. A detailed description of each of these benchmark functions is given in [20].

MPCE & SSALS performed 25 times for each benchmark function. All tests were completed using MATLAB R2019a. The dimension D of all functions is 1000, except that f13 and f14 are 905. The stopping criterion was a fixed number of fitness evaluations (FEs). The Max_NFE was set to be 3.0E+6, and the program terminates when Max_NFE is reached. The initial value of NP was set to 100, CP1 = 0.3, CP2 = 0.29, CP3 = 0.29, I_{NPD} = 100, and I_{GS} = 40. The statistical results including the best, the worst, the median, the mean, and the standard deviation computed over 25 runs are shown in Table 1.

4.1. Influence of the Different Components. In this section, experiments were conducted to observe the influence of the different components. For each test, Table 2 lists the average results of 25 independent runs. Wilcoxon signed-rank test with the significance level of 5% was used for statistical analysis. The “>” “<”, and “=” mean “significantly better,” “significantly worse,” and “no significant difference,” respectively. The last row of Table 2 shows the times of win/tie/loss (w/t/l) in the pairwise comparison.

To observe the individual effect of both Multiparent Crossover Evolution and Step-Size Adaptive Local Search, experiments were executed on the two algorithms, respectively. As shown in Table 2, the optimization performance of SSALS is significantly better than that of MPCE on most of the functions, indicating that local searches contribute more to the hybrid algorithm.

The influence of the number of parents was also studied. In the proposed algorithm, CP2 and CP3 are the crossover probability constants of parent 2 and parent 3, respectively. CP2 = 0 indicates that parent 2 does not participate in the evolutionary operation, so is CP3. In one test, CP2 was set to 0, indicating that the three-parent crossover operation was used. In another test, CP2 and CP3 were both set to 0, which means that a two-parent crossover operation was applied. The other settings are considered the same as in Table 1. According to the results, increasing the number of parents affects f5, f6, f9, and f10, but it has no significant difference on other functions. In general, it is beneficial and harmless.

In addition, to verify the improvement effect of SSASL, the same experiments were performed in identical conditions with replacing MTS-LS1 with SSALS. In the comparison test between SSALS with MTS-LS1, it is clear that SSALS is significantly better than MTS-LS1 on all 15 functions, demonstrating that the optimization performance of SSALS is significantly higher than that of MTS-LS1.

4.2. Parameter Analysis. Major parameters of the MPCE & SSALS are I_{NPD} and I_{GS}. For each I_{NPD} iteration, population size (NP) is subtracted from 1. When NP is reduced to 4, the population-based search ends, which means only the individual-based search is performed to improve the best solution. Therefore, a smaller I_{NPD} means fewer population searches and more single individual searches. I_{GS} indicates the number of iterations between two global searches, and a smaller I_{GS} represents more global searches and fewer local searches. Different I_{NPD} and I_{GS} are studied in this section. MPCE & SSALS performed 25 times for each combination. Wilcoxon signed-rank test was used for statistical analysis.

To find the appropriate I_{NPD} and I_{GS} value, three CEC’2013 benchmark functions f3, f7, and f15 are studied in the tests with I_{NPD} value varying from 50 to 500 and I_{GS} value varying from 20 to 200, respectively. The other parameters use the same settings as in Table 1. Algorithm performances by adopting different I_{NPD} and I_{GS} values on f3, f7, and f15 are shown in Figures 1(a), 1(b), and 1(c), and Figures 2(a), 2(b), and 2(c), respectively. The horizontal axis represents the respective parameter settings while the vertical axis shows the obtained logarithm value of mean FEs. As shown in Figures 1 and 2, the best result was obtained when I_{NPD} = 100 and I_{GS} = 40.

The test results of some different combinations of I_{NPD} and I_{GS} with {50,40}, {100,40}, {200,40}, {100,20}, and {100,80} in 15 benchmark functions are shown in Table 3. Results shown in bold indicate the final selected parameters. When fixing I_{GS} value at 40, I_{NPD} = 100 significantly outperforms I_{NPD} = 50 in 5 functions and is outperformed by I_{NPD} = 50 in 1 function, while other the 9 functions have no significant difference. I_{NPD} = 100 significantly outperforms I_{NPD} = 200 in 7 functions and is outperformed by I_{NPD} = 50 in 2 functions, which indicate that the best overall optimization performance is I_{NPD} = 100. When I_{NPD} is fixed at 100, I_{GS} = 40 significantly outperforms I_{GS} = 20 and I_{GS} = 80 in 2 and 3 functions, respectively. There is no significant difference in other functions, which means that the best optimization performance is I_{GS} = 40. As shown in Table 3, MPCE-SSALS with I_{NPD} = 100 and I_{GS} = 40 significantly outperforms the other parameter settings. It is also observed that the best parameter values of different test functions may be different; for example, the best parameter for f3 and f6 is I_{NPD} = 200 and I_{GS} = 40. This suggests that it is a good choice to set the parameter values to 100 and 40 in general, but for a specific problem, better parameter values can be determined by experiments.

In addition, when I_{NPD} = 100 and I_{GS} = 40, the FEs of global search and local search are 12524 and 2987476, respectively, which indicates that the proposed algorithm is mainly based on local search.

4.3. Comparison with the Reference Algorithm. MPCE & SSALS was compared with other state-of-the-art algorithms, including SHADE-ILS [12], MLSHADE-SPA [5], CBCC-RDG3 [17], and IMLSHADE-SPA [11]. Among them, SHADE-ILS, CBCC-RDG3, and MLSHADE-SPA were currently the top three algorithms in the CEC LSGO competitions, and IMLSHADE-SPA is an improved version of MLSHADE-SPA.

To compare these algorithms’ performances on the CEC’2013 function suite, the average ranking of each algorithm was calculated. For a fair comparison, the other four algorithms’ experimental data and supplementary material
Table 2: Comparison of different components on the CEC 2013 LSGO functions, $D = 1000$, $\text{FEs} = 3.0E + 06$ (with Wilcoxon test, $\alpha = 0.05$).

| Fun | MPCE | SSALS | MPCE & SSALS | MPCE & SSALS | MPCE & MTS-LS1 | MPCE & SSALS |
|-----|------|-------|--------------|--------------|----------------|--------------|
|     | Four parents | Two parents | Three parents | Four parents | Four parents | Four parents |
| f1  | 3.50E + 10  | 1.53E - 27  | 1.48E - 27  | 2.49E - 27  | 3.24E - 25  | 1.53E - 27  |
| f2  | 2.63E + 04  | 5.97E - 01  | 1.52E + 00  | 1.39E + 00  | 3.28E + 02  | 1.35E + 00  |
| f3  | 2.01E + 01  | 1.32E + 01  | 3.42E - 13  | 3.59E - 13  | 1.65E - 12  | 3.59E - 13  |
| f4  | 1.03E + 12  | 3.67E + 09  | 3.51E + 09  | 3.70E + 09  | 1.52E + 10  | 3.56E + 09  |
| f5  | 6.75E + 06  | 2.24E + 07  | 1.45E + 06  | 1.31E + 06  | 2.00E + 06  | 1.18E + 06  |
| f6  | 6.43E + 05  | 9.84E + 05  | 5.44E + 04  | 3.98E + 03  | 2.53E + 05  | 7.46E + 02  |
| f7  | 1.74E + 10  | 1.29E + 05  | 6.07E + 03  | 5.20E + 03  | 1.30E + 07  | 4.39E + 03  |
| f8  | 2.86E + 16  | 3.14E + 14  | 1.60E + 14  | 1.32E + 14  | 1.46E + 15  | 1.84E + 14  |
| f9  | 5.97E + 08  | 1.52E + 09  | 1.13E + 08  | 9.60E + 07  | 1.50E + 08  | 9.02E + 07  |
| f10 | 1.86E + 07  | 8.90E + 07  | 1.03E + 05  | 6.51E + 04  | 8.14E + 06  | 4.98E + 04  |
| f11 | 4.42E + 12  | 1.15E + 07  | 3.76E + 06  | 3.65E + 06  | 1.41E + 10  | 3.32E + 06  |
| f12 | 7.66E + 11  | 4.98E + 01  | 9.40E + 01  | 4.45E + 01  | 8.07E + 02  | 5.85E + 01  |
| f13 | 1.61E + 12  | 4.06E + 07  | 2.52E + 06  | 2.19E + 06  | 1.19E + 09  | 3.56E + 06  |
| f14 | 4.99E + 12  | 7.65E + 06  | 5.35E + 06  | 5.51E + 06  | 1.65E + 10  | 5.46E + 06  |
| f15 | 4.83E + 13  | 8.58E + 05  | 5.93E + 05  | 5.76E + 05  | 4.26E + 07  | 5.80E + 05  |

| w/t/l | 15/0/0 | 11/3/1 | 4/11/0 | 2/13/0 | 15/0/0 | / |

![Figure 1](image1.png)

**Figure 1:** Comparison of different $I_{NPD}$ values. (a) $f_3$. (b) $f_7$. (c) $f_15$.

![Figure 2](image2.png)

**Figure 2:** Comparison of different $I_{GS}$ values. (a) $f_3$. (b) $f_7$. (c) $f_15$.

are directly taken from their original papers. Wilcoxon signed-rank test (significance level = 0.05) is utilized for pairwise comparison of these five algorithms. The place of each algorithm on each function is calculated according to the Wilcoxon rank tests. The comparison results and ranking are listed in Table 4. The best results for each benchmark function are distinguished by bold font. As shown in Table 4, MPCE & SSALS is the one with the best performance among these algorithms. SHADE-ILS and IMLSHADE-SPA rank second and third, respectively.
Table 3: Optimization results of MPCE-SSALS with different combinations of parameters $I_{\text{NPD}}$ and $I_{\text{GS}}$ (with Wilcoxon test, $\alpha = 0.05$).

| Fun | $I_{\text{NPD}} = 100$, $I_{\text{GS}} = 40$ | $I_{\text{NPD}} = 50$, $I_{\text{GS}} = 40$ | $I_{\text{NPD}} = 200$, $I_{\text{GS}} = 40$ | $I_{\text{NPD}} = 100$, $I_{\text{GS}} = 20$ | $I_{\text{NPD}} = 100$, $I_{\text{GS}} = 80$ |
|-----|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|
| f1  | $1.53E-27$  | $5.95E-28$  | $1.22E-27$  | $5.78E-28$  | $2.15E-27$  |
| f2  | $1.35E+00$  | $4.98E-01$  | $3.48E+00$  | $1.09E+00$  | $1.31E+00$  |
| f3  | $3.59E-13$  | $3.80E-13$  | $3.61E-13$  | $3.62E-13$  | $3.67E-13$  |
| f4  | $3.56E+09$  | $3.14E+09$  | $5.12E+09$  | $3.83E+09$  | $3.41E+09$  |
| f5  | $1.18E+06$  | $1.43E+06$  | $1.20E+06$  | $1.17E+06$  | $1.51E+06$  |
| f6  | $7.46E+02$  | $5.36E+03$  | $3.52E+02$  | $7.15E+02$  | $4.93E+03$  |
| f7  | $4.39E+03$  | $3.97E+03$  | $1.31E+04$  | $5.46E+03$  | $5.61E+03$  |
| f8  | $1.84E+14$  | $1.58E+14$  | $1.91E+14$  | $1.74E+14$  | $1.58E+14$  |
| f9  | $9.02E+07$  | $1.13E+08$  | $8.83E+07$  | $8.47E+07$  | $1.11E+08$  |
| f10 | $4.98E+04$  | $1.42E+05$  | $2.08E+04$  | $4.03E+04$  | $8.12E+04$  |
| f11 | $3.32E+06$  | $1.25E+06$  | $6.65E+06$  | $4.61E+06$  | $5.06E+06$  |
| f12 | $5.85E+01$  | $3.20E+01$  | $2.71E+02$  | $1.45E+02$  | $6.90E+01$  |
| f13 | $3.56E+06$  | $2.06E+06$  | $1.09E+07$  | $2.34E+06$  | $4.06E+06$  |
| f14 | $5.46E+06$  | $5.30E+06$  | $7.70E+06$  | $5.45E+06$  | $5.47E+06$  |
| f15 | $5.80E+05$  | $5.55E+05$  | $7.01E+05$  | $6.19E+05$  | $6.01E+05$  |
| w/t/l | — | 5/9/1 | 7/6/2 | 2/13/0 | 3/12/0 |

Table 4: Experimental comparisons between MPCE & SSALS and state-of-the-art algorithms on the CEC’2013 LSGO functions, $D = 1000$, $FEs = 3.0E + 06$ (with Wilcoxon test, $\alpha = 0.05$).

| Fun | MPCE & SSALS | SHADE-ILS | MLSHADE-SPA | CBCC-RDG3 | IMLSHADE-SPA |
|-----|--------------|-----------|-------------|-----------|--------------|
|     | Mean (std)   | Place     | Mean (std)  | Place     | Mean (std)   | Place     |
| f1  | 1.53E-27     | 1.34E-02  | 2.69E-24    | 1.94E-22  | 1.14E-18     | 4.0E-04   |
| f2  | 1.35E+00     | 2.14E-02  | 1.00E+03    | 7.89E+01  | 2.31E+03     | 4.65E+03  |
| f3  | 3.59E-13     | 1.24E-02  | 2.01E+01    | 9.69E+00  | 2.04E+01     | 2.28E+00  |
| f4  | 3.56E+09     | 1.48E-02  | 1.48E+08    | 6.90E+08  | 4.29E+04     | 3.78E+08  |
| f5  | 1.18E+06     | 1.39E-02  | 1.39E+06    | 1.80E+06  | 2.04E+06     | 1.75E+06  |
| f6  | 7.46E+02     | 2.14E-02  | 1.02E+06    | 1.40E+03  | 1.00E+06     | 3.67E+03  |
| f7  | 4.39E+03     | 7.41E+01  | 7.41E+01    | 5.31E+04  | 1.71E+21     | 4.90E+03  |
| f8  | 1.84E+14     | 3.17E+11  | 3.17E+11    | 9.77E+12  | 7.11E+03     | 5.40E+12  |
| f9  | 9.02E+07     | 1.64E+08  | 1.64E+08    | 1.61E+08  | 1.57E+08     | 1.63E+08  |
| f10 | 4.98E+04     | 9.18E+07  | 9.18E+07    | 6.36E+02  | 9.16E+07     | 4.95E+02  |
| f11 | 3.32E+06     | 5.11E+05  | 5.11E+05    | 4.04E+07  | 2.18E-13     | 2.00E+06  |
| f12 | 5.85E+01     | 1.04E+02  | 1.04E+02    | 1.04E-02  | 7.00E-02     | 1.01E-02  |
| f13 | 3.56E+06     | 7.19E+04  | 7.19E+04    | 7.19E+04  | 6.43E+04     | 3.05E+06  |
| f14 | 5.46E+06     | 5.76E+06  | 5.76E+06    | 5.76E+06  | 6.43E+04     | 2.11E+06  |
| f15 | 5.80E+05     | 2.40E+05  | 2.76E+07    | 9.01E+06  | 2.30E+06     | 1.23E+06  |
| w/t/l | — | 8/2/5 | 9/2/4 | 10/9/0 | 7/5/3 |

Avg R of f1–f3  | 1.33  | 3.00  | 2.00  | 4.33  |
Avg R of f4–f11 | 2.94  | 2.88  | 3.44  | 2.63  |
Avg R of f12–f15 | 2.25  | 2.13  | 4.13  | 3.88  |
Avg R           | 2.43  | 2.70  | 3.33  | 3.30  |
Ranking         | 1     | 2     | 5     | 4     | 3     |
Figure 3: Continued.
Compared with SHADE-ILS, MLSHADE-SPA, CBCC-RDG3, and IMLSHADE-SPA, MPCE & SSALS won in 8, 9, 10, and 7 functions but lost in 5, 4, 5, and 3 functions, respectively. For the 15 benchmark functions, MPCE & SSALS obtained 8 best results, while SHADE-ILS, MLSHADE-SPA, CBCC-RDG3, and IMLSHADE-SPA obtained 3, 4, 5, and 3 best results, respectively. For fully separable functions, MPCE & SSALS performs the best. CBCC-RDG3 achieves the best on partially separable functions. For overlapping functions and fully nonseparable functions, MPCE & SSALS and SHADE-ILS give the best results.

Figure 3 shows the convergence curve of MPCE & SSALS, SHADE-ILS, MLSHADE-SPA, CBCC-RDG3, and IMLSHADE-SPA in 15 benchmark functions.

As shown in the figure, the results could be summarized as follows:

(i) MPCE & SSALS has a faster convergence rate than IMLSHADE-SPA in the functions f1–f3, f5–f7, f9, and f13–f15, but it has a slower convergence rate than IMLSHADE-SPA in the functions f4, f8, and f10–f12.

(ii) MPCE & SSALS has a faster convergence rate than SHADE-ILS in the functions f2, f3, f5, f6, f9, f10, and f15, but it has a slower convergence rate than SHADE-ILS in the functions f1, f4, f7, f8, and f11–f14.

(iii) MPCE & SSALS has a faster convergence rate than MLSHADE-SPA in f1, f5, f6, f7, f9, and f13–f15 functions; however, it has a slower convergence rate than MLSHADE-SPA in f3, f4, f8, f10, and f12 functions.

(iv) MPCE & SSALS has a faster convergence rate than CBCC-RDG3 in f2, f3, f6, f10, and f15 functions, but it has a slower convergence rate than CBCC-RDG3 in f4, f7, f8, f11, and f13 functions.

The convergence rate of MPCE & SSALS, in general, is faster than that of MLSHADE-SPA and IMLSHADE-SPA, but it has a similar convergence rate to CBCC-RDG3 and SHADE-ILS. However, MPCE & SSALS is the simplest one among these algorithms.

4.4. Results Discussion. The excellent results of MPCE & SSALS mainly benefit from the following factors:

(1) The multiparent strategy used in this paper enables each offspring to inherit genes from multiple excellent individuals. It not only increases the offspring diversity, but also moves the algorithm quicker towards better solutions.

(2) SSALS effectively improves MTS-LS1, which enhanced the local search performance significantly. Each dimension has its own basic step size that can be adjusted to accommodate the different effects of each dimension on the function. In addition, the minimum step size affects the search accuracy. If a large number of high-precision searches (very small step size) are carried out in the early stage of the algorithm, it will waste an amount of calculation and easily fall into the local minima. According to the current search results, gradual improvement to the search accuracy can avoid excessive search in the early stage of the algorithm. Similarly, in the late stage of the algorithm, the most promising position can be searched with high precision.

(3) More local searches are performed in the algorithm. The proposed algorithm performs much more local searches than global searches. This enhances the exploitation capability of the algorithm in the search space.

(4) Population decrease strategy is used in MPCE & SSALS. At the beginning of the algorithm, a large population size is conducive to improving the exploration ability. With the optimization process, individual differences are minimized, and the advantages of a large population are also reduced. Gradually reducing the population size is helpful to enhance the exploitation ability.

(5) The memetic algorithm framework is used to combine the multiparent strategy, SGSO and SSALS, to work together. The memetic algorithm framework balances the exploration ability of global search and...
the exploitation ability of local search; thus, it has been widely used in LSGO problems. The SGSO algorithm also performs well in LSGO problems.

5. Conclusions
In this paper, a memetic algorithm MPCE & SSALS based on multiparent crossover evolution and step-size adaptive local search is proposed for LSGO problem. The MPCE strategy is used for global exploration, and the SSALS method is applied for local exploitation. In the early stage of algorithm execution, the global search and the local search are performed interchangeably, and the population size is gradually reduced to 1. In the later stage, only the local search is executed to improve the final solution. Local search is performed during the whole process, and the execution times of local search is far more than that of global search. A set of 15 benchmark functions was used to evaluate the performance of the MPCE & SSALS algorithm. According to the experimental data, the overall performance of the MPCE & SSALS algorithm performs better than the other four state-of-the-art algorithms. The experimental results also indicate that the performance of SSALS is significantly higher than that of MTS-LS1, and the local search-dominated hybrid algorithm can effectively solve the LSGO problem.

On the other hand, the experiment analysis reveals that the multiparent crossover strategy can only improve the optimization effect of certain test functions while having no discernible impact on others. Among the four parents in the crossover operation, three individuals are selected from the previous generation of the population. The source of parents is relatively single. The advantages of multiparenting were not fully used. In the future, it is possible to add new parent generation methods, such as using PSO to generate one of the parents. This paper demonstrated that multiparent crossover evolution combined with local search is an effective algorithm framework to address the LSGO problem. A possible extension to this paper is to examine new parent generation techniques or local search strategies that improve the algorithm’s performance.

Data Availability
The source code and experimental data of MPCE & SSALS can be requested from yydzhwf@xnu.edu.cn.

Conflicts of Interest
The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments
This work was supported in part by the Hunan Education Department Scientific Research Key Project under Grant 17A201 and in part by the Science and Technology Development Program of Chenzhou under Grant 201810212.

References
[1] F. Zhao, L. Zhang, Y. Zhang, W. Ma, C. Zhang, and H. Song, “A hybrid discrete wave optimization algorithm for the no-idle flowshop scheduling problem with total tardiness criterion,” Expert Systems with Applications, vol. 146, Article ID 113116, 2020.
[2] S. Zhou, L. Xing, Z. Zheng, N. Du, L. Wang, and Q. Zhang, “A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times,” IEEE Transactions on Cybernetics, vol. 51, no. 3, pp. 1430–1442, 2021.
[3] F. Zhao, L. Zhao, L. Wang, and H. Song, “An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion,” Expert Systems with Applications, vol. 160, Article ID 113678, 2020.
[4] W.-L. Liu, Y.-J. Gong, W.-N. Chen, Z. Liu, H. Wang, and J. Zhang, “Coordinated charging scheduling of electric vehicles: a mixed-variable differential evolution approach,” IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 12, pp. 5094–5109, 2020.
[5] A. A. Hadi, A. W. Mohamed, and K. M. Jambi, “LSHADE-SPA memetic framework for solving large-scale optimization problems,” Complex & Intelligent Systems, vol. 5, no. 1, pp. 25–40, 2018.
[6] W. Guo, L. Zhu, L. Wang, Q. Wu, and F. Kong, “An entropy-assisted particle swarm optimizer for large-scale optimization problem,” Mathematics, vol. 7, no. 5, p. 414, 2019.
[7] F. Kong, J. Jiang, and Y. Huang, “An adaptive multi-swarm competition particle swarm optimizer for large-scale optimization,” Mathematics, vol. 7, no. 6, p. 521, 2019.
[8] H. Deng, L. Peng, H. Zhang, B. Yang, and Z. Chen, “Ranking-based biased learning swarm optimizer for large-scale optimization,” Information Sciences, vol. 493, pp. 120–137, 2019.
[9] A. Latorre, S. Muelas, and J.-M. Pena, “Large scale global optimization: experimental results with MOS-based hybrid algorithms,” in Proceedings of the 2013 IEEE Congress on Evolutionary Computation, pp. 2742–2749, Cancun, MX, USA, June 2013.
[10] L. Y. Tseng and C. Chen, “Multiple trajectory search for large scale global optimization,” in Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pp. 3052–3059, Hong Kong, China, June 2008.
[11] H. G. Koçer and S. A. Uymaz, “A novel local search method for LSGO with golden ratio and dynamic search step,” Soft Computing, vol. 25, no. 3, pp. 2115–2130, 2020.
[12] D. Molina, A. LaTorre, and F. Herrera, “SHADE with iterative local search for large-scale global optimization,” in Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1252–1259, Rio de Janeiro, Brazil, July 2018.
[13] W. Liu, Y. Zhou, B. Li, and K. Tang, “Cooperative Co-evolution with soft grouping for large scale global optimization,” in Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), pp. 319–325, Wellington, New Zealand, June 2019.
[14] L. Li, W. Fang, Q. Wang, and J. Sun, “Differential grouping with spectral clustering for large scale global optimization,” in Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), pp. 443–446, Wellington, New Zealand, June 2019.
[15] Z. Ren, A. Chen, M. Wang, Y. Yang, Y. Liang, and K. Shang, “Bi-hierarchical cooperative coevolution for large scale global optimization,” IEEE Access, vol. 8, pp. 41913–41928, 2020.
M. Yang, A. Zhou, C. Li, J. Guan, and X. Yan, “CCFR2: a more efficient cooperative co-evolutionary framework for large-scale global optimization,” Information Sciences, vol. 312, pp. 64–79, 2020.

Y. Sun, X. Li, A. Ernst, and M. N. Omidvar, “Decomposition for large-scale optimization problems with overlapping components,” in Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), pp. 326–333, Wellington, New Zealand, June 2019.

H. Liu, Y. Wang, L. Liu, and X. Li, “A two phase hybrid algorithm with a new decomposition method for large scale optimization,” Integrated Computer-Aided Engineering, vol. 25, no. 4, pp. 349–367, 2018.

A. Vakhnin and E. Sopov, “An approach for initializing the random adaptive grouping algorithm for solving large-scale global optimization problems,” IOP Conference Series: Materials Science and Engineering, vol. 537, no. 4, Article ID 042006, 2019.

X. Li, K. Tang, M. Omidvar, Z. Yang, and K. Qin, “Benchmark functions for the CEC2013 special session and competition on large scale global optimization,” Evolutionary Computation and Machine Learning Group, Technical Report, RMIT University, Melbourne, Australia, 2013.

M. Mauˇcec 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 and Evolutionary Computation, vol. 50, Article ID 100428, 2019.

A. Singh and N. Dulal, “A survey on metaheuristics for solving large scale optimization problems,” International Journal of Computer Application, vol. 170, no. 5, pp. 1–7, 2017.

P. Moscato, “On evolution, search, optimization, genetic algorithms and martial arts - towards memetic algorithms,” Caltech Concurrent Computation Program, Technical Report 826, California Institute of Technology, Pasadena, CA, USA, 1989.

W.-f. Zhang, “Simplified group search optimizer algorithm for large scale global optimization,” Journal of Shanghai Jiaotong University, vol. 20, no. 1, pp. 38–43, 2015.

C. Cotta, L. Mathieson, and P. Moscato, “Memetic algorithms, in Handbook of HeuristicsSpringer, New York, NY, USA, 2018.

R. Mencia, M. R. Sierra, C. Mencia, and R. Varela, “Memetic algorithms for the job shop scheduling problem with operators,” Applied Soft Computing, vol. 34, pp. 94–105, 2015.

W. Yi, X. Li, and B. Pan, “Solving flexible job shop scheduling using an effective memetic algorithm,” International Journal of Computer Applications in Technology, vol. 53, no. 2, pp. 157–163, 2016.

H. Samma, C. P. Lim, and J. Mohamad Saleh, “A new reinforcement learning-based memetic particle swarm optimizer,” Applied Soft Computing, vol. 43pp. 276–297, C, 2016.

L. Tan, J. Sun, and X. Tong, “A hybrid particle swarm optimization based memetic algorithm for DNA sequence compression,” Soft Computing, vol. 19, no. 5, pp. 1255–1268, 2015.

J.-C. Hung, “Memetic particle swarm optimization scheme for direction-of-arrival estimation in multipath environment,” Journal of Intelligent and Fuzzy Systems, vol. 34, no. 6, pp. 3955–3968, 2018.

D. Jia, T. Li, Y. Zhang, and H. Wang, “A memetic artificial bee colony algorithm for high dimensional problems,” International Journal of Computational Intelligence and Applications, vol. 19, no. 01, Article ID 2050008, 2020.

Y. Chen and J.-K. Hao, “Memetic search for the generalized quadratic multiple knapsack problem,” IEEE Transactions on Evolutionary Computation, vol. 20, no. 6, pp. 908–923, 2016.

V. A. Shim, K. C. Tan, and H. Tang, “Adaptive memetic computing for evolutionary multiobjective optimization,” IEEE Transactions on Cybernetics, vol. 45, no. 4, pp. 610–621, 2015.

P. T. H. Nguyen and D. Sudholt, “Memetic algorithms outperform evolutionary algorithms in multimodal optimisation,” Artificial Intelligence, vol. 287, Article ID 103345, 2020.

Y. Jin and J.-K. Hao, “Solving the Latin square completion problem by memetic graph coloring,” IEEE Transactions on Evolutionary Computation, vol. 23, no. 6, pp. 1015–1028, 2019.

A. W. Mohamed, A. A. Hadi, A. M. Fatouhou, and K. M. Jambi, “LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems,” in Proceedings of the. 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 145–152, Donostia, Spain, June 2017.

A. A. Hadi, A. W. Mohamed, and K. M. Jambi, “Single Objective Real-Parameter Optimization: Enhanced Lshade-Spaca Algorithm,” Heuristics for Optimization and Learning, Springer, New York, NY, USA, 2021.

X. Li and S. Ma, “Multi-objective memetic search algorithm for multi-objective permutation flow shop scheduling problem,” IEEE Access, vol. 4, pp. 2154–2165, 2016.

Z. Zhang, Y. Sun, H. Xie, Y. Teng, and J. Wang, “GMMA: GPU-based multiobjective memetic algorithms for vehicle routing problem with route balancing,” Applied Intelligence, vol. 49, no. 1, pp. 63–78, 2019.

R. Tanabe and A. Fukunaga, “Evaluating the performance of SHADE on CEC 2013 benchmark problems,” in Proceedings of the 2013 IEEE Congress on Evolutionary Computation, pp. 1952–1959, Cancun, Mexico, June 2013.

A. W. Mohamed, “Solving large-scale global optimization problems using enhanced adaptive differential evolution algorithm,” Complex & Intelligent Systems, vol. 3, no. 4, pp. 205–231, 2017.

A. W. Mohamed and A. S. Almazyad, “Differential evolution with novel mutation and adaptive crossover strategies for solving large scale global optimization problems,” Applied Computational Intelligence and Soft Computing, Article ID 7974218, 2017.

A. LaTorre, A framework for hybrid dynamic evolutionary algorithms: multiple offspring sampling (MOS), Ph.D. Dissertation, Facultad de Informática, Universidad Politécnica de Madrid, Madrid, ESP, 2009.

A. LaTorre, S. Muelas, and J.-M. Peña, “A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test,” Soft Computing, vol. 15, no. 11, pp. 2187–2199, 2011.

A. LaTorre, S. Muelas, and J.-M. Peña, “Multiple offspring sampling in large scale global optimization,” in Proceedings of the. 2012 IEEE Congress on Evolutionary Computation, pp. 964–971, Brisbane, QLD, Australia, June 2012.

F. J. Solis and R. J.-B. Wets, “Minimization by random search techniques,” Mathematics of Operations Research, vol. 6, no. 1, pp. 19–30, 1981.

Z. H. Ahmed, “A multi-parent genetic algorithm for the quadratic assignment problem,” Opsearch, vol. 52, no. 4, pp. 714–732, 2015.

A. E. Eiben, P.-E. Raué, and Z. Ruttkay, “Genetic algorithms with multi-parent recombination,” in Proceedings of the Parallel Problem Solving from Nature - PPSN III, pp. 78–87, Jerusalem, Israel, October 1994.

S. P. T. Pann Phyu and G. Srijuntongsiri, “Effect of the number of parents on the performance of multi-parent genetic algorithm,” in Proceedings of the 2016 11th International Conference on Knowledge, Information and Creativity Support
[50] S. Tsutsui, M. Yamamura, and T. Higuchi, “Multi-parent recombination with simplex crossover in real coded genetic algorithms,” in Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation, pp. 657–664, New York, NY, USA, January 1999.

[51] A. Arram and M. Ayob, “A novel Multi-parent order crossover in Genetic Algorithm for combinatorial optimization problems,” Computers & Industrial Engineering, vol. 133, pp. 267–274, 2019.

[52] S. He, Q. H. Wu, and J. R. Saunders, “A novel group search optimizer inspired by animal behavioural ecology,” in Proceedings of the 2006 IEEE International Conference on Evolutionary Computation, pp. 4415–4421, Vancouver, BC, Canada, July 2006.

[53] R. Tanabe and A. S. Fukunaga, “Improving the search performance of SHADE using linear population size reduction,” in Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 1658–1665, Beijing, China, July 2014.